

Программно-аппаратные платформы Интернета вещей и встраиваемые системы

Лекция 13

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ

Что делать с данными?

- Практический пример: фитнес-трекер
- Оцифровка ЭКГ – 16 бит, 1000 отсчетов/с
- Передаваемые данные – 10 байт/с (ЧСС, RR-интервалы)



Цифровая обработка сигналов

- Фильтры
 - БИХ, бесконечная импульсная характеристика (IIR, infinite impulse response)
 - КИХ, конечная импульсная характеристика (FIR, finite impulse response)
- Преобразование Фурье
- ...и другие подобные алгоритмы

Что есть в Cortex-M?

- Cortex-M0 – базовый набор команд, 32-битный умножитель
- Cortex-M3 – добавляются инструкции типа multiply-accumulate (MAC), 32-битное деление (2-12 тактов)
- Cortex-M4 – SIMD-инструкции, арифметика с насыщением
 - Опционально, в Cortex-M4F – арифметика с плавающей запятой, обычной точности (IEEE754 single precision)
- Cortex-M7 – в опциональном модуле арифметики с плавающей запятой добавляется повышенная точность (double precision)

Пример: multiply-accumulate

```
int64_t dot_product (int32_t *x, int32_t *y, int32_t N) {  
    int32_t xx, yy, k;  
    int64_t sum = 0;  
    for(k = 0; k < N; k++) {  
        xx = *x++;  
        yy = *y++;  
        sum += xx * yy;  
    }  
    return sum;  
}
```

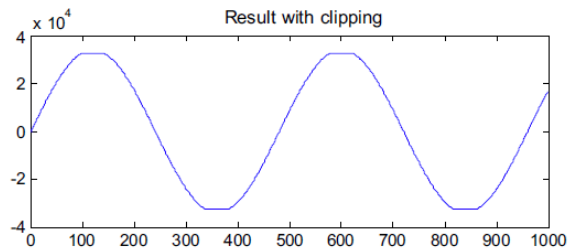
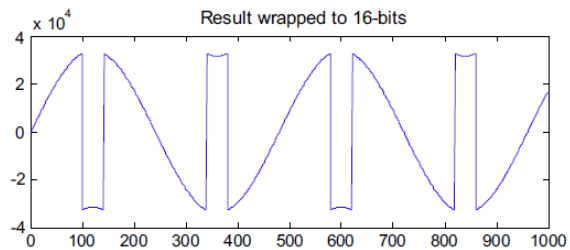
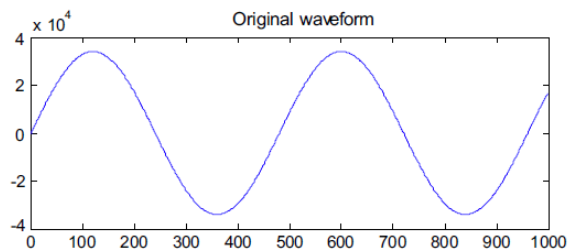
На Cortex-M3

```
int64_t dot_product (int32_t *x, int32_t *y, int32_t N) {  
    int32_t xx, yy, k;  
    int64_t sum = 0;  
    for(k = 0; k < N; k++) { // цикл [3 такта]  
        xx = *x++; // [2 такта]  
        yy = *y++; // [2 такта]  
        sum += xx * yy; // [от 3 до 7 тактов]  
    }  
    return sum;  
}
```

На Cortex-M4

```
int64_t dot_product (int32_t *x, int32_t *y, int32_t N) {  
    int32_t xx, yy, k;  
    int64_t sum = 0;  
    for(k = 0; k < N; k++) { // цикл [3 такта]  
        xx = *x++; // [2 такта]  
        yy = *y++; // [1 такт] (load после load)  
        sum += xx * yy; // MAC [1 такт]  
    }  
    return sum;  
}
```


Арифметика с насыщением

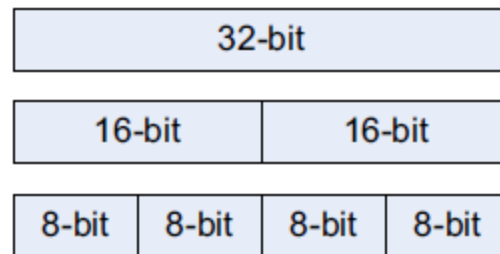


Форматы чисел q15, q31

- Способ представления дробных чисел в диапазоне от -1 до $1-2^{-N}$, иногда пишут $[-1, 1)$
- Арифметика полностью повторяет таковую для целых чисел
- Надо держать в голове размерности величин, с которыми приходится работать
- Ограниченная в сравнении с float точность, обнуление малых значений
- bit-exactness, независимость от архитектуры

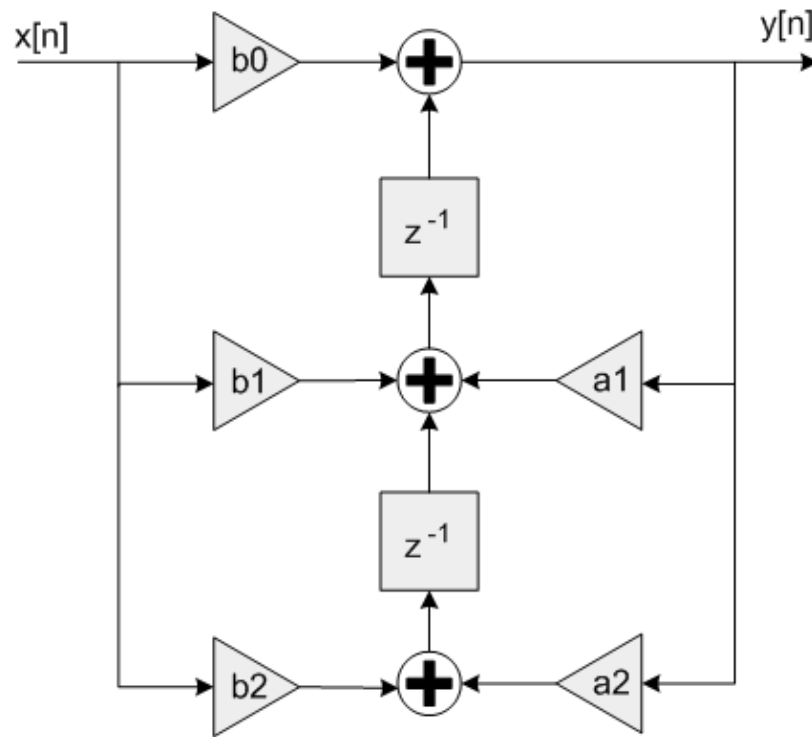
SIMD-инструкции

- Single instruction, multiple data
- Оперируем одним регистром, в который загружено несколько значений
- Обычная арифметика – сложение, умножение, multiply-accumulate



Пример: БИХ-фильтр

- $y_n = b_0x_n + (b_1x_{n-1} + a_1y_{n-1}) + (b_2x_{n-2} + a_2y_{n-2})$
- 5 multiply-accumulate и три переменных состояния



Запишем на C (для float)

```
// b0, b1, b2, a1, и a2 – коэффициенты фильтра
// stateA, stateB, и stateC – состояние фильтра
for (sample = 0; sample < blockSize; sample++) {
    stateA = *inPtr++ + a1*stateB + a2*stateC;
    *outPtr++ = b0*stateA + b1*stateB + b2*stateC;
    stateC = stateB;
    stateB = stateA;
}
```

Как выглядит цикл на уровне инструкций?

```
stateA = *inPtr++;    // Загрузка в регистр [2 такта]
stateA += a1*stateB;  // MAC [4 такта]
stateA += a2*stateC;  // MAC [4 такта]
out = b0*stateA;      // Умножение [2 такта]
out += b1*stateB;     // MAC [4 такта]
out += b2*stateC;     // MAC [4 такта]
*outPtr++ = out;      // Сохранение в память [2 такта]
stateC = stateB;      // Перемещение между регистрами [1 такт]
stateB = stateA;      // Перемещение между регистрами [1 такт]
// Цикл [3 такта]
// Итого 27 тактов
```

Следующий шаг

```
stateA = *inPtr++; // Загрузка в регистр [2 такта]
prod1 = a1*stateB; // Умножение [1 такт]
prod2 = a2*stateC; // Умножение [1 такт]
stateA += prod1; // Сложение [1 такт]
prod4 = b1*stateB; // Умножение [1 такт]
stateA += prod2; // Сложение [1 такт]
out = b2*stateC; // Умножение [1 такт]
prod3 = b0*stateA // Умножение [1 такт]
out += prod4; // Сложение [1 такт]
out += prod3; // Сложение [1 такт]
stateC = stateB; // Перемещение между регистрами [1 такт]
stateB = stateA; // Перемещение между регистрами [1 такт]
*outPtr++ = out; // Сохранение в память [2 такта]
// Цикл [3 такта]
// Итого 18 тактов
```

Дальнейшая оптимизация

- Loop unrolling
 - Вместо цикла пишем его тело «ручками»
- Переиспользуем переменные (точнее, это регистры) для stateA, stateB, stateC в таком порядке:
 - ABC
 - CAB
 - BCA
- Загрузку данных выносим перед циклом
- Это сократит выполнение трех итераций с $18 \times 3 = 54$ тактов до 38 (12,67 тактов на отсчет)

CMSIS-DSP

Библиотека CMSIS-DSP

- Оптимизированная (пример выше взят оттуда) реализация для ARM Cortex-M алгоритмов ЦОС и математических функций «общего назначения»
 - Векторная арифметика
 - Тригонометрические функции
 - Операции с матрицами
 - БИХ и КИХ фильтры
 - Преобразования Фурье
- Большинство функций реализовано для 32-битных float, q15 и q31

Пример: БИХ-фильтр

- Данные от АЦП накапливаются в буфере
- При заполнении буфера данные начинают писаться во второй буфер, а в первом фильтруются

Пример: БИХ-фильтр

```
q31_t in_buf[512];
q31_t out_buf[512];

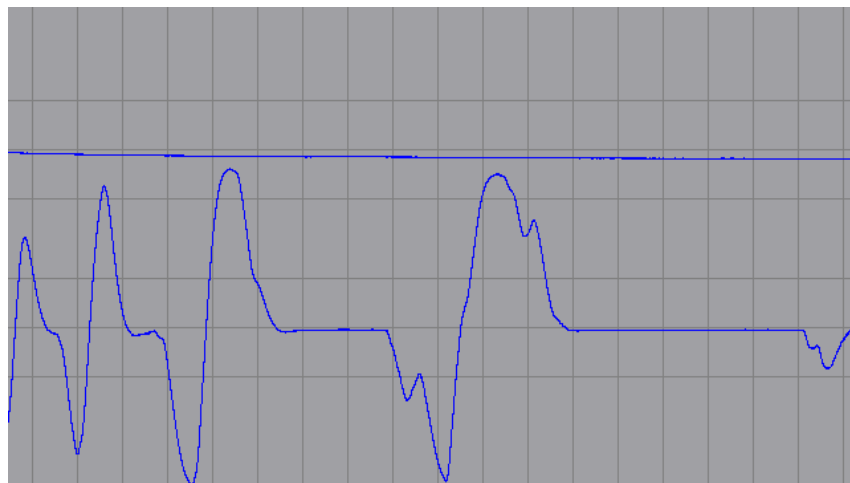
q31_t filter_coeffs[STAGES*5]; /* 5 coefficients for each stage */
q63_t filter_state[STAGES*4]; /* 4 state variables for each stage */
arm_biquad_cas_df1_32x64_ins_q31 filter;

arm_biquad_cas_df1_32x64_init_q31(&filter,
                                  STAGES,
                                  filter_coeffs,
                                  filter_state,
                                  0);

arm_biquad_cas_df1_32x64_q31(&filter,
                              in_buf,
                              out_buf,
                              num_samples);
```

Пример: БИХ-фильтр

- 32-битный фильтр с 32-битными переменными состояния
- ФВЧ Чебышева 2 порядка
- Частота оцифровки – около 1000 Гц, частота среза 0,1 Гц



Немного модного ML

- В CMSIS-DSP реализован SVM-классификатор
- CMSIS-NN – отдельная библиотека для вычислений с использованием сверточных нейросетей
 - Нет возможности обучения, нейросеть готовится отдельно, а затем полученные коэффициенты конвертируются в понятный CMSIS-NN формат

CMSIS-NN - производительность

- Задача классификации изображений из датасета CIFAR-10 (32*32)
- Сверточная сеть из 7 слоев, работает на Cortex-M7@216 МГц
- Около 10 изображений/с

