КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

E.P. Sosnenko

COMPARATIVE ANALYSIS OF CONSTRUCTION OF THE DECODER OF THE CONVOLUTION CODE

We consider the application and features of decoding convolution codes. A comparative analysis of the construction of the Viterbi decoder in an environment of Windows, Assebler on DSP and FPGA.
Key words: Convolutional codes, Viterbi Algorithm.

Розглядаються області застосування й особливості декодування кодів, які згортаються. Виконано порівняльний аналіз побудови декодера Вітербі в середовищі ОС Windows, на Асеблері цифрового сигнального процесора і в ПЛІС FPGA.
Ключові слова: коди, які згортаються, алгоритм Вітербі.

Рассмотрены области применения и особенности декодирования сверточных кодов. Выполнен сравнительный анализ построения декодера Витерби в среде ОС Windows, на Ассеблере цифрового сигнального процессора и в ПЛИС FPGA.
Ключевые слова: сверточные коды, алгоритм Витерби.

УДК 681.3(031)

Е.П. СОСНЕНКО

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СРЕД РЕАЛИЗАЦИЙ ДЕКОДЕРА СВЕРТОЧНОГО КОДА

Введение. Сверточные коды нашли широкое применение в телекоммуникационных системах цифровой связи (протоколы V.32, V.34, ADSL, HDSL) и интегрированных производственных системах управления для помехоустойчивого кодирования передаваемой информации по каналам связи с шумами.

Декодирование сверточных, а в последнее время и турбо-кодов – процесс несравненно более затратный и длительный по сравнению с кодированием входных сообщений. В настоящее время для этих целей используется алгоритм Витерби, обеспечивающий максимально правдоподобную оценку равновероятных входных кодовых слов.

Необходимость в сравнительном анализе программных и аппаратных средств декодирования сверточных кодов возникла в связи с разработкой методов полунатурного моделирования устройств и процессов в системах управления и аппаратуре специального назначения, проектируемой на ПЛИС совместно с другими функциональными узлами.

Следует отметить, что в настоящее время пропускная способность каналов связи вплотную приблизилась к границе, определенной Шенноном и в основном ограничена сложностью процессов декодирования сообщений.

Все вышесказанное определяет актуальность выполненных исследований по повышению эффективности декодирования сверточных кодов и тематику рассматриваемой работы.

Из всех известных методов декодирования сверточных кодов наиболее эффективным по совокупности функциональных характеристик

и затрат является декодер на основе алгоритма Витерби. Декодер Витерби реализует метод максимального правдоподобия и оптимизирован с учетом использования конечной структуры используемой конкретной решётки кода.

Кодирование сверточных кодов производится с помощью $k0$ сдвиговых регистров (по числу информационных входов кодера), имеющих $n0$ выходов, образованных сумматорами по *mod*2 состояний части разрядов регистров. В каждом такте работы кодер воспринимает $k0$ входных информационных символов, и выдает на выход $n0$ выходных символов, подлежащих передаче по каналу связи. Выходные символы, в каждом такте зависят от информационных символов, поступивших на этом и на $n - 1$ предыдущих тактах, где n – длина сдвигового регистра (далее рассматриваются случаи $k0 = 1$). Отношение $k0/n0$ называется скоростью сверточного кода, а $CL = n - 1$ – его кодовым ограничением.

Именно длина регистра в первую очередь определяет сложность декодера сверточного кода и его корректирующие способности, а выбранное конечное ограничение этой длины допускает эффективное использование декодера Витерби.

Помимо скорости, сверточный код (а именно, подключаемые на выход через сумматоры или напрямую разряды регистра) задается порождающими полиномами, количеством $n0$ не менее двух. Краткая запись в восьмеричной форме через запятые векторов бинарных коэффициентов полиномов представляет его формат. Например, широко описываемый (5,7)-код имеет два выхода сумматоров по *mod*2, первый из которых связан с нулевым и вторым, а второй – со всеми тремя разрядами трехразрядного регистра сдвига (регистр не должен иметь по краям не используемых "висячих" разрядов).

Работу декодера Витерби удобно рассматривать с помощью решетчатой диаграммы, представляющей циклически повторяемый участок более громоздкой формы описания кода – дерева кода. На рис. 1 показана решетчатая диаграмма, динамически генерируемая в рабочей среде, разработанной специально для моделирования процессов кодирования и декодирования сверточных кодов с помощью алгоритма Витерби.

Решетчатая диаграмма состоит из ярусов (вертикальных колонок) по числу входных $n0$-битовых информационных символов. В каждом ярусе представлены 2^{n-1} номеров узлов, соответствующие всем возможным состояниям n-1 крайних левых разрядов регистра кодера при передаче очередного и пока еще не известно какого именно входного символа. Ребра решетчатой диаграммы указывают на возможные переходы регистра кодера при его сдвиге на один разряд вправо и соответствующие им переходы алгоритма (решетки) декодера из одного узла-источника в другой узел-приемник.

Двоичный номер $An–1\ An–2\ ...\ A2\ \ A1$ соединенного ребрами решетчатой диаграммы узла-источника и номер $An\ An–1\ An–2\ ...\ A2$ узла-приемника связаны общим для них состоянием $n–2$ (без двух крайних правых) разрядов регистра кодера $An–1\ An–2\ .....\ A2$:

$$An–1\ An–2\ ....\ A2\ \ A1 \quad\ A0 \qquad (1)$$
$$An\ \ An–1\ .....A3\ \ A2 \quad\ A1 \quad \text{сдвиг вправо.}$$

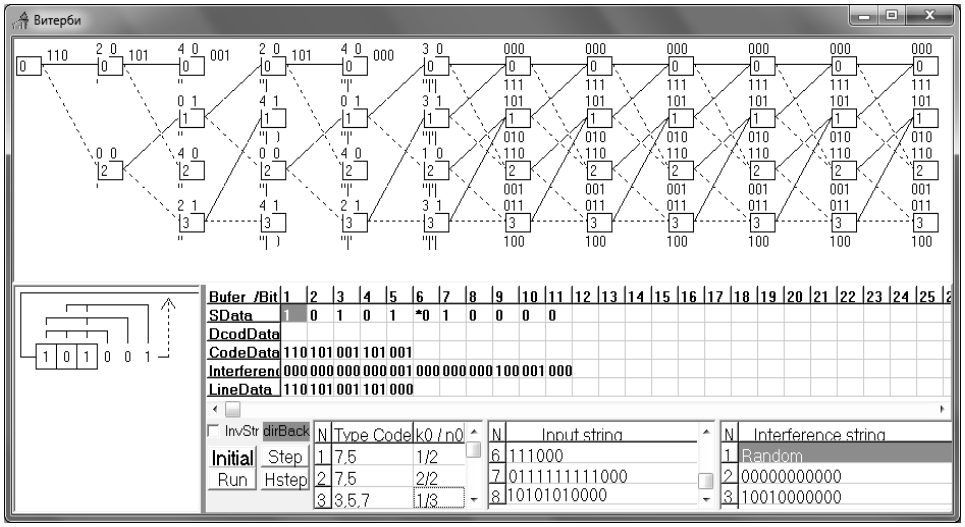

РИС. 1. Рабочая среда для моделирования процессов кодирования и декодирования сверточных кодов

Эти 2^{n-2} жестко заданных условия определяют 2^{n-2} связанных групп узлов, каждая из которых состоит из двух узлов-источников и двух узлов-приемников (по аналогии с быстрым преобразование Фурье – "бабочка" Витерби).

Переходы из узлов одной "бабочки" в узлы другой в пределах яруса запрещены условием (1). Это ограничение на полносвязность графа решетки эффективно используется декодером Витерби. Внутри "бабочки" доступны четыре конфигурации переходов – для каждого из двух приемников возможны два узла-источника. При этом номер узла-источника дополняется до полных *n*-1 разрядов соответственно битом *A*1 в крайнем правом разряде регистра кодера, не влияющем на нумерацию узлов. Номер узла-приемника дополняется до полного битом *An*, представляющим входной информационный бит кодера.

Ни *A*1 ни *An* не известны на приемной стороне, только известна избыточная по сравнению с единственным входным битом *An*, но возможно ошибочная входная кодовая комбинация *InLine* из *n*0 бит, образованная порождающими полиномами кодера и помехами в линии связи. Декодер Витерби сравнивает вектор *InLine* с векторами четного *Rib*0 и нечетного *Rib*1 ребер, хранящимися в памяти декодера (или генерируемыми порождаемыми полиномами, как функции от возможных состояний *An An*–1 .....*A*3 *A*2 *A*1). Суммы отличающихся битов представляют собой расстояния Хемминга между векторами и называются метриками ребер, соответственно *MRb*0 и *MRb*1.

При переходе к очередному ярусу решетки декодер подсчитывает метрики путей *MTr*0, *MTr*1, суммируя предыдущие их значения с метриками ребер *MRb*0, *MRb*1. При этом в соответствии с основным принципом динамического про-

граммирования исключается подпуть с большей метрикой из двух, ведущих в узел, поскольку он не может служить префиксом наилучшего пути от этого узла.

Можно показать, что при выборе в качестве выжившего, подпути с минимальной метрикой *MTr*, определяемой суммой метрик ребер *MRb,* т. е. суммой расстояний Хемминга, декодер Витерби реализует выбор оптимального пути в соответствии с методом максимального правдоподобия [*].

Изложенные выше особенности алгоритма декодера Витерби позволяют уменьшить число обращений к памяти за значениями *MTr* и *Trk*, особенно в случае аппаратной реализации алгоритма, что показано в его блок-схеме на рис. 2.

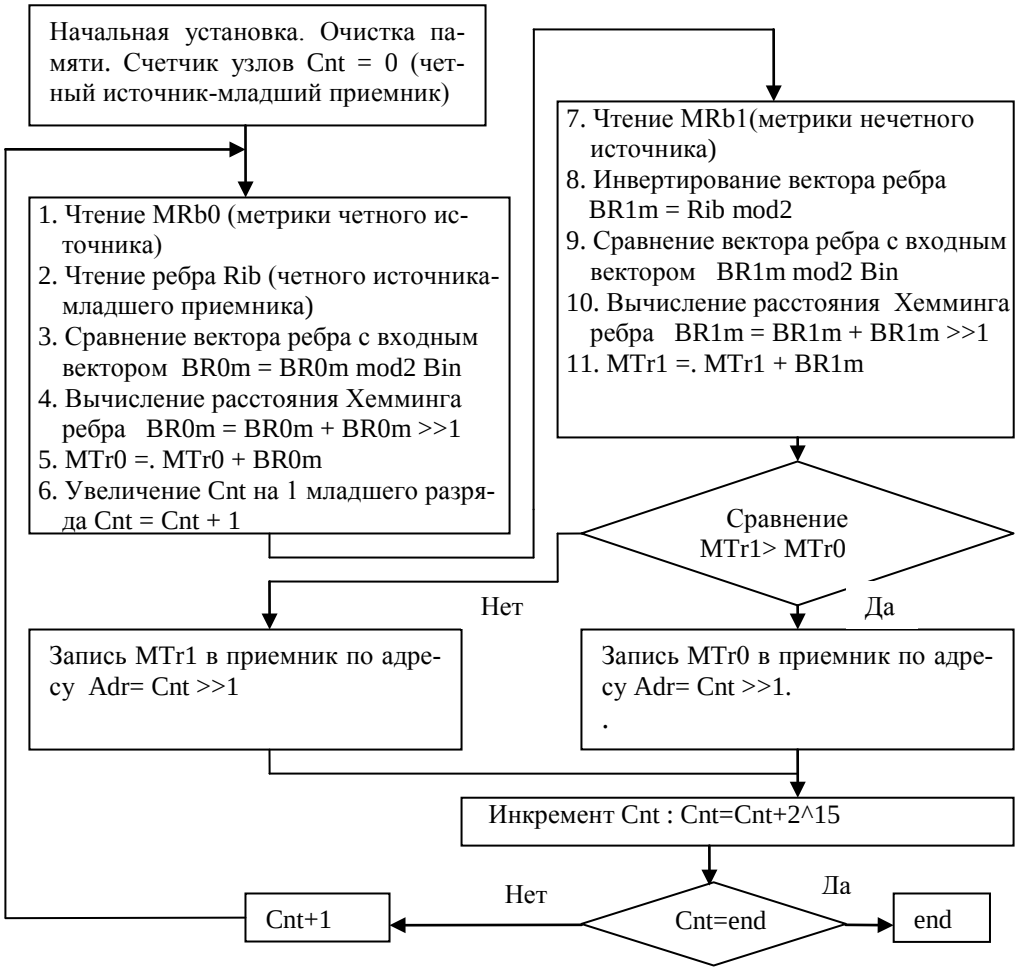

РИС. 2. Блок-схема алгоритма декодера Витерби

* *Скляр Б.* Цифровая связь. Теоретические основы и практическое применение, 2-е изд.: Пер. с анг. – М.: Издательский дом "Вильямс", 2003. – 1104 с.

Алгоритм содержит два крупных блока вычисления и сравнения метрик четных *MTr*0 и нечетных *MTr*1 путей для младшего узла-приемника, выполняемых последовательно или параллельно в зависимости от возможностей программной и аппаратной его реализации. После увеличения счетчика номеров узлов на единицу старшего разряда выполняется циклический переход к аналогичной обработке метрик для старшего узла-приемника.

В зависимости от результатов сравнения в обоих шагах цикла выполняется запись в память метрик и треков обновленных значений метрик *MTr* и треков *Trk* путей. Новое значение *Trk* образуется путем сдвига выжившего прежнего значения с добавлением в освободившийся разряд нулевого - для младшего и единичного - для старшего приемника значения *An*. Обработка очередного яруса решетки заканчивается при достижении счетчиком узлов конечного значения $LM = 2^{n-1}$, а выполнение алгоритма - при достижении конечного нулевого узла (при выдаче в линию терминальной последовательности нулей длиной равной кодовому ограничению сверточного кода).

Рассмотрим пример программной реализация обработки яруса решетки на высокоуровневом языке в среде ОС Windows:

```
HalfLM:=LM shr 1;   CNode:=0;              // CNode – счетчика адреса узлов
While   CNode <  LM - 1  do  begin         // цикл по "бабочкам"яруса решетки
Mtr0:=Metric[CNode];      Trk0:=  Track[CNode];
CNode:= CNode+1;                           // адрес старшего источника
Mtr1:=Metric[CNode];      Trk1:=  Track[CNode];
CNode:=CNode  shr 1;                       // адрес младшего приемника
Rib:= Ribr[CNode];
 for  j:=0   to  1  do   begin  //цикл по младшему и старшему приемнику
MRb0:= Rib  xor  InLine;   MRb0:= MRb0 + (MRb0  shr 1);
MRb0:= MRb0 + Mtr[0];
Rib:= Rib xor 3;              //получение инверсного вектора единичного ребра
MRb1:= Rib  xor  InLine;   MRb1:= MRb1 + (MRb1  shr 1);
MRb1:= MRb1 + Mtr[1];
 if  MRb0 <= MRb1
   then begin Metric[CNode]:=MRb0; Track [CNode]:=(Trk0 shl 1) or j;  end
   else begin Metric[CNode]:=MRb1;  Track [CNode]:=(Trk1 shl 1) or j;  end;
CNode:=CNode  xor HalfLM; end;             // адрес старшего приемника
CNode:=(CNode shl 1) +2;        end;       // переход к следующей "бабочке"
```

Программа содержит дважды выполняемый цикл для обновления значений метрик и треков в узлах младшего и старшего приемников с последовательным вычислением метрик ребер и итоговых метрик путей для четного и нечетного источника в каждом проходе цикла. Выделенные команды обработки счетчика узлов *CNode* изменяют состояние его младшего разряда при чтении источников и разряда на единицу меньшего длине регистра кодера (с предварительным делением значения *CNode на 2*) соответственно при записи приемников.

Вектора ребер привязаны к адресам узлов-приемников. Элементы управления дублированием памяти не показаны.

Функционально аналогичный участок кода на ассемблере цифрового сигнального процессора (DSP), отличающегося развитой системой двух- и трехместной адресации и возможностью выполнения в одной команде до двух комплекснозначных или скалярных операции приведен далее.

```
// DSP *** VITERBI ***********************************************
    // Обработка 2-х метрик конкурирующих путей младшего приемника
L0.1: M"[0]=B"[jB] xor T"[jT];                  // сравн.2-х.вект.рёб.и вход.вект.
    C"[4]=  P"[jB];                             // чт.путей(от четн.и нечетн.ист)
    C"[0]= M"[0]s + M"[0];                      // подсчет раст. Хеминга 2-х рёбер
    C"[0]= C"[0] + N"[jB],   jB=jB+sB;          //суммир. метрик 2-х конкур. путей
    R= C[0] - C [1], if R>0 goto L0.2; //переход, если мет.чет.ист.> мет.нечет.ист.
    N[jB] = C[1];                   //сохр. в Мл.прием. суммар. метр. нечет. источ.
    P[jP]=C[5]s+1,  goto L0.3;      //сохр. в Мл.прием. обновл. трека  нечет. источ.
L0.2: N[jB] = C[0];                 //сохр. в Мл.прием. суммар. метр. четного ист.
    P[jP]=C[4]s;                    //сохр. в Мл.прием. обновл. трека чет. источ.
    // Обработка 2-х метрик конкур. путей старшего приемника (аналогично)
L0.3: B"[jB]= B"[jB]  shr 1,  jP=jP-sP+1, if  jP< lP goto  L0.1;  //цикл по ярусу
    T"[jT]=T"[jT] shr 1, jP=jP+1, jT=jT+sT,  if  jT< lT goto  L0.1; //следующ. ярус
```

Код выгодно отличается от вышерассмотренного, прежде всего:

- параллельной обработкой метрик ребер и итоговых суммарных метрик четного и нечетного узлов-источников за счет возможности одновременного выполнения двух и более скалярных операций в одной команде DSP;

- отсутствием команд модификации и обработки значений счетчика узлов, так как в DSP функции адресно-индексной обработки используют встроенный блок базовых, индексных и дополнительных регистров, обеспечивающих параллельное упреждающее вычисление адресов операндов с помощью независимых адресно-индексных генераторов;

- отсутствием выделенных команд условных и безусловных переходов, что в ряде случаев позволяет уменьшить размер и время выполнения программы.

В полной мере достичь высокой производительности работы декодера сверточных кодов по алгоритму Витерби позволяет его построение программируемых ПЛИС FPGA, например, Spartan3 фирмы Xilinx. Показанная на рис. 3 блок-схема декодера может быть выполнена в четырехтактном или двухтактном варианте с реализацией всей обработки метрик после их чтения из блочной памяти ПЛИС с помощью комбинационной схемы. При этом затраты площади кристалла ПЛИС меньше затрат на один адресный генератор DSP.

В таблице приведены характеристики рассмотренных вариантов реализации декодера Витерби с CL=11 (2048 состояний) в средах с различным сочетанием в них программной и аппаратной составляющих.

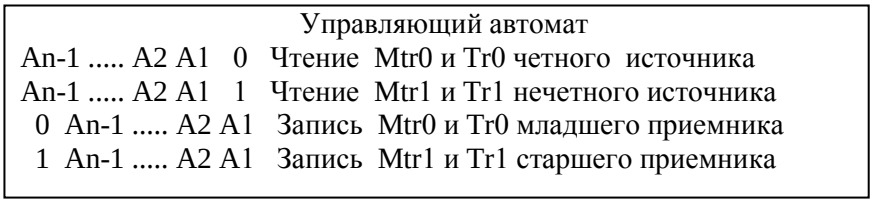

РИС. 3. Блок-схема декора сверточного кода

ТАБЛИЦА

Среда реализации	Число тактов обработки "бабочки"	Время выполнения такта (нс)	Время обработки символа (мкс)
PC, ОС Windows	150 – 250	0,5 – 2	75 – 500
DSP, FPGA	15	50 – 500	750 –7500
Декодер Витерби FPGA	2 – 4	10 – 50	20 – 200

Выводы. Конкретный выбор той или иной среды должен определяться в контексте дополнительных требований. Так для встроенных систем целесообразно применять часть кристалла FPGA для построения высокопроизводительного варианта декодера в окружении других функциональных блоков.

Получено 02.10.2015