КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

УДК 681.3(031)

Е.Н. ЧИЧИРИН

ПЕРЕСТРАИВАЕМАЯ СТРУКТУРА ДЕКОДЕРА ВИТЕРБИ В БАЗИСЕ ПЛИС XILINX

Основной резерв в повышении эффективности управления интегрированными системами управления предприятиями заключен в организации высокоэффективного обмена данными на участке сети с наиболее насыщенным трафиком, т. е. между ERP, MES и SCADA системами. Есть два пути решения этой задачи:

- повышение эффективности средств связи между указанными подсистемами;
- организация распределенной интеллектуальной обработки в иерархической системе, созданной программно-аппаратными средствами на ERP, MES и SCADA уровнях.

При решении обеих задач используется алгоритм Витерби, как основной метод динамического программирования, обеспечивающий максимальное правдоподобие принимаемых решений. Интеллектуальная обработки «сырых» данных необходима для выявления существенных переменных и паттернов протекающих процессов. Результатом такого предварительного анализа будет трансляция наверх наиболее значимой информации при сохранении прежних объемов трафика и возможность ее использования в локальных PLC-системах контурного регулирования. Наиболее востребованными методами локальной обработки являются:

- деревья принятия решений;
- нейро-сетевые алгоритмы;
- визуализация многомерных данных;
- методы, использующие скрытые марковские модели (СММ).

E.N. Chichirin

RECONFIGURABLE THE VITERBI DECODER ON THE BASIS OF FPGA XILINX

It is considered the areas of the Viterbi algorithm and its implementation in basis FPGA Xilinx. The proposed solutions allow to work with different formats of convolutional codes.
Key words: Convolutional codes, Viterbi Algorithm.

Розглянуто області застосування алгоритму Вітербі та його реалізація в базисі ПЛІС Xilinx. Запропоновані рішення забезпечують роботу з різними форматами кодів, які згортаються.
Ключові слова: коди, які згортаються, алгоритм Вітербі.

Рассмотрены области применения алгоритма Витерби и его реализация в базисе ПЛИС Xilinx. Предлагаемые решения обеспечивают работу с различными форматами сверточных кодов.
Ключевые слова: сверточные коды, алгоритм Витерби.

Перечисленные методы составляют базис методов анализа данных Data Mining и находят все большее применение для создания адаптивных и самоорганизующихся алгоритмов цифровой обработки сигналов, в том числе в информационных системах управления производством [1]. Обработка первичных данных с помощью перечисленных алгоритмов предусматривает выделение из них наиболее значимых комбинаций или паттернов (например, наиболее часто повторяющихся сочетаний) с целью идентификации и последующей обработки для принятия решений. В конечном итоге, выделенная информация необходима для вышестоящих производственных служб, представленных планировочными и аналитическими приложениями MES и ERP-систем.

Задача разработки методов и средств моделирования межуровневой интеграции систем управления производством может быть эффективно решена при условии, что математическая модель будет адекватна процессам, которые протекают в такой интегрированной системе.

В силу неполноты информации о первичных составляющих таких процессов особый интерес представляет использование методов, составляющих основу скрытых марковских моделей (СММ), а именно:

- алгоритм Витерби – делает наилучшее предположение о последовательности состояний скрытой модели на основе последовательности наблюдений. Эта последовательность состояний называется путем Витерби;

- алгоритм Баума–Уэлча (или EM-алгоритм – метод математического ожидания-модификации), обеспечивающий обучение модели так, чтобы она как можно лучше описывала реальную наблюдаемую последовательность. Задача обучения СММ – это важнейшая задача для большинства проектируемых СММ, и заключается в оптимизации их параметров на основе обучающей последовательности;

- алгоритм "вперед-назад", находящий вероятность попадания в скрытое состояние на очередном шаге процесса и входит, как составная часть в вышеперечисленные алгоритмы.

Широкое применение алгоритм Витерби находит также как метод декодирования сверточных кодов, применяемых в современных системах цифровой связи. Этот алгоритм реализует декодирование по критерию максимального правдоподобия и обеспечивает минимальную вероятность ошибки при равновероятных кодовых словах [2, 3].

Все вышесказанное обусловило необходимость повышения эффективности методов и средств реализации алгоритма Витерби при решении различных задач в современных системах связи и управлении производством. В этом плане представляется актуальной разработка аппаратных средств поддержки для декодирования сверточных кодов и решения задач СММ.

Применение в качестве аппаратной среды доступных в настоящее время перепрограммируемых больших интегральных схем типа FPGA открывает большие возможности объединения в одном кристалле средств для решения помимо вышеназванных также и связанных с ними задач управления производством.

Обзор мировых производителей микросхем FPGA показал, что наиболее оптимальным выбором следует признать перепрограммируемые кристаллы фирмы Xilinx, как наиболее динамично совершенствующуюся продукцию, удовлетворяющую основным требованиям как по функциональным, так и по стоимостным характеристикам. В особенности это относится к последней 7-й серии микросхем FPGA: Artix, Kintex и Vertex. С основными характеристиками FPGA фирмы Xilinx можно ознакомиться в [4], а также на сайте Xilinx.com.

К сожалению, стартовые затраты на приобретение отладочных средств и несильно урезанной версии САПР достаточно велики. Поэтому значительную часть возможностей FPGA , в частности, использование блочной памяти в двухпортовом режиме пришлось отрабатывать теоретически или с помощью программного моделирования на персональном компьютере.

Рабочая среда для моделирования процессов кодирования и декодирования сверточных кодов по алгоритму Витерби реализована с использованием средств высокоуровневоого программирования и отличается развитыми возможностями визуализации всех процессов обработки информации, в том числе, в пошаговом режиме. Рабочее окно среды показано на рис. 1.

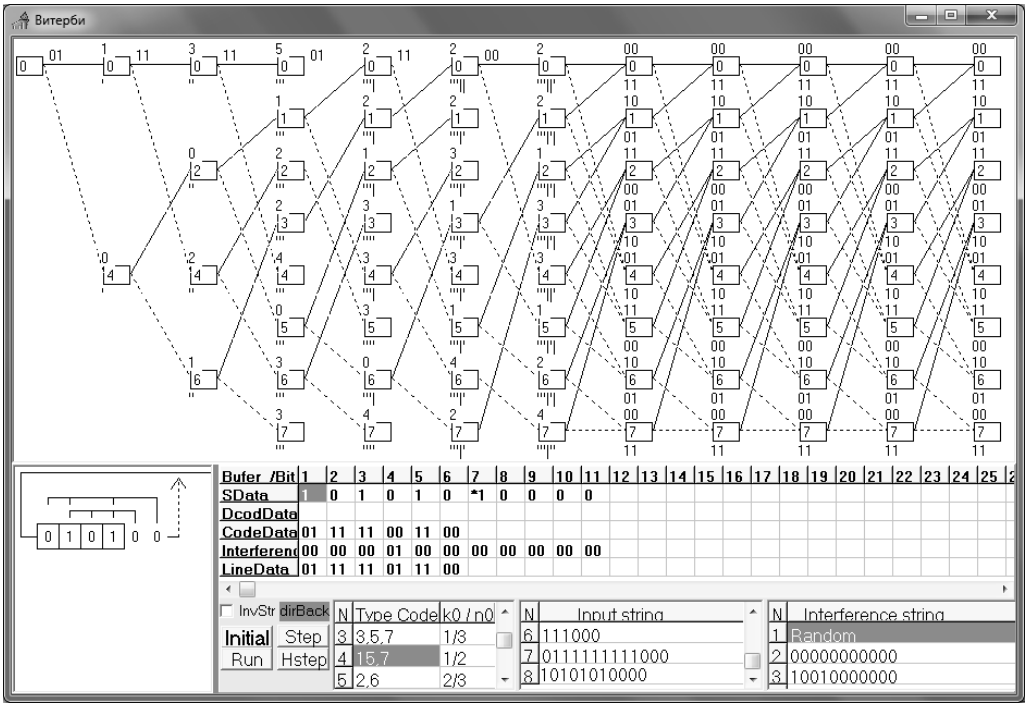

РИС. 1. Рабочая среда для моделирования кодера и декодера сверточных кодов

В верхней части окна расположена модель-диаграмма, отражающая пути прохождения алгоритма декодирования по узлам решетки с индикацией метрик конкурирующих и выбывших путей и накапливаемых в прямом проходе принятых кодов.

Внизу слева направо расположены:

- функциональная модель кодера сверточного кода со сдвиговыми регистрами согласно заданной скорости **k0/n0** и соответствующей структурой выбранных полиномов;
- таблица состояний с динамически обновляемыми после обработки каждого входного символа, а также после завершения обработки всей входной последовательности, строками исходных данных **SData**, декодированных данных **DcodData**, данных на выходе кодера **CodeData**, помех в линии связи **Interference**, и данных на входе декодера **LineData**;
- переключатель прямого или инверсного порядка отображения ребер путей на решетчатой диаграмме **InvStr**;
- переключатель однопроходного **Direct**, двухпроходного **DirectBack** или комбинированного **Comby** вариантов алгоритма декодирования;
- клавиша инициализации **Initial** структуры решетчатой диаграммы, кодера и таблицы состояний в соответствии с выбранными форматом сверточного кода, входной информационной строкой и строкой помех в лини связи (в том числе сгенерированными случайным образом);
- клавиши непрерывного **Run**, пошагового **Step** и полушагового **Hstep** (независимого для кодера и декодера) исполнения алгоритма;
- таблица выбора скорости **k0/n0** и формата (полиномов) кода **TypeCode;**
- таблица выбора входной последовательности **Input String**;
- таблица выбора последовательности помех в линии связи **Iinterference String**.

Среда позволяет проводить моделирование процессов кодирования сверточных кодов с кодовым ограничением CL от 2 до 16, количеством входов **k0** и выходов **n0** соответственно до 2 и 3.

Чтобы найти ближайшее к принятой последовательности кодовое слово, мы движемся по решетке слева направо, оставляя на входе каждого узла лишь один подпуть – ближайший к соответствующему префиксу принятой последовательности. Мы исключаем худший подпуть из двух, ведущих в каждый узел, поскольку в соответствии с принципом динамического программирования он не может служить префиксом наилучшего в дальнейшем пути от этого узла. В случае, если оба подпути находятся на одинаковом расстоянии от соответствующего префикса принятой последовательности, мы используем правило равновероятного разрешения неопределенности (coin-flip tie-breaking rule). Мы продолжаем двигаться слева направо до тех пор, пока не достигнем конечного нулевого узла (при выдачи в линию терминальной последовательности нулей длиной равной кодовому ограничению сверточного кода). Поскольку выживает лишь один путь на решетке, он и является наилучшим. Задержка декодирования равна длине кодового слова, поскольку окончательное решение не может быть принято до

тех пор, пока все узлы на данной глубине не будут иметь один и тот же начальный подпуть.

В терминах принятых в рабочей среде обозначений алгоритм Витерби находит кодовое слово **CodeData = CodeData*,** максимизирующее апостериорную вероятность P(**CodeData | LineData**) передачи исходной кодовой последовательности **CodeData** при условии, что принята последовательность **LineData**. Для цифровых каналов алгоритм сводится к нахождению последовательности **CodeData = CodeData*,** ближайшей к **LineData** в смысле расстояния Хэмминга dH **(LineData, CodeData),** т. е. **CodeData * =** argmin **{dH(LineData, CodeData)}.**

Исследования, проведенные при моделировании в рабочей среде процессов кодирования и декодирования сверточных кодов позволили сделать следующие выводы:

- двухкратное снижение числа тактов чтения метрик и треков (входных последовательностей или обратных указателей) из оперативной памяти (RAM) достигается при совместной обработке двух узлов приемников, имеющих общие для них два узла источника ("бабочка" Витерби). Это условие является также необходимым для отказа от дублирования памяти, но требует усложнения блока формирования адресов и будет рассмотрено в дальнейшем;

- метрики ребер между источниками и приемниками могут генерироваться аппаратно либо храниться в памяти в силу относительно небольших затрат по сравнению затратами на память метрик и тем более треков;

- однопроходный алгоритм декодирования **Direct** с учетом постоянно снижающейся стоимостью RAM предпочтительнее двухпроходного **DirectBack** как более простой и производительный. Для длинных входных последовательностей возможен комбинированный метод **Comby** с организацией обратных указателей на большие не перезаписываемые отрезки входных последовательностей.

С другой стороны, анализ FPGA фирмы Xilinx показал:

- сравнительно небольшие затраты на аппаратный вычислитель метрик ребер позволяют организовать параллельный подсчет, компарирование и мультиплексирование метрик и треков двух источников на входах блочной RAM;

- двухпортовая блочная память дает возможность существенно (до двух раз) ускорить процесс декодирования, отказаться от буферных регистров метрик и треков и упростить цепи синхронизации;

- типовая структура ячеек LUT не дает возможность достаточно эффективно использовать площадь и ресурсы кристалла FPGA.

Одним из основных требований, предъявляемым при разработке декодера, являлась возможность его оперативного перепрограммирования, а не переконфигурирование под различные форматы сверточных кодов. Минимаксная стратегия (а именно, минимум затрат на максимальную конфигурацию) дополнялась тактической оптимизацией отдельных элементов и узлов, в частности разделением адреса RAM и кода вычислителя метрик ребер на две части [5].

Блок-схема декодера Витерби показана на рис. 2. Декодер может быть представлен в виде контроллера адресов и контроллера данных блоков памяти метрик и треков.

Источником входных кодов для обеих подсхем является генератор (счетчик) номеров узлов, осуществляющий последовательный перебор адресов двух узлов источников и двух узлов приемников каждой "бабочки" очередного яруса решетчатой диаграммы.

Контроллер адресов представляет собой общий для обоих блоков памяти мультиплексор адресной шины, управляемый выходом второго разряда генератора узлов.

Контроллер данных содержит независимые мультиплексоры шин данных для каждого из блоков памяти, управляемые выходом компаратора блока обработки метрик треков.

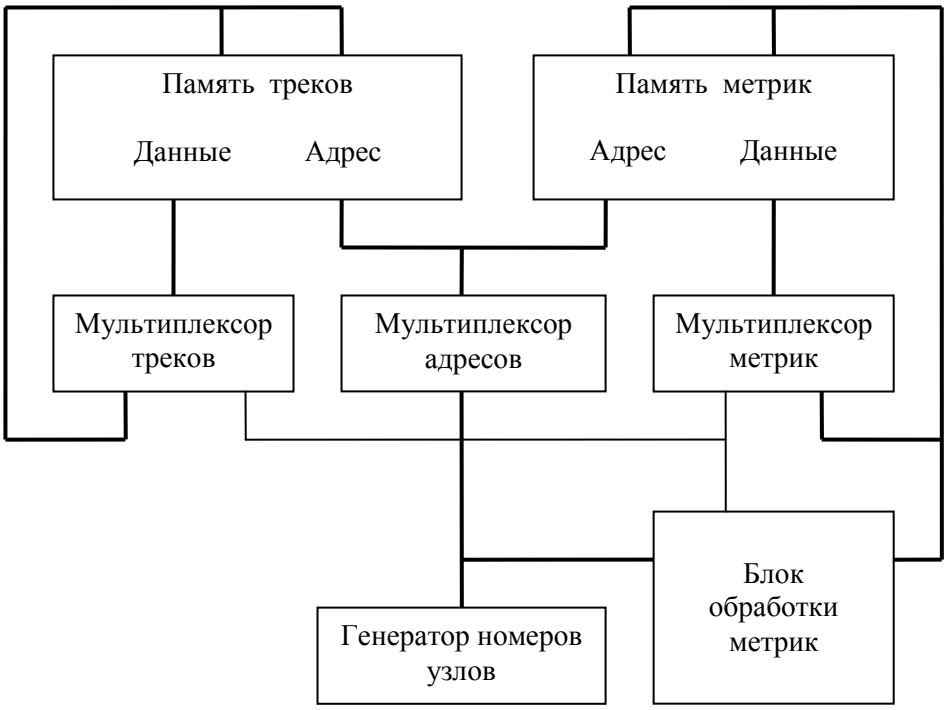

РИС. 2. Блок-схема декодера Витерби

Блок обработки метрик треков содержит схемы формирования метрик ребер между двумя узлами источниками и двумя узлами приемниками внутри каждой "бабочки", вычисления и сравнения итоговых метрик треков для каждого из двух узлов приемников.

Алгоритм работы декодера Витерби при обработке очередной "бабочки", представлен в таблице.

ТАБЛИЦА

Такт	Код на входах адреса RAM	Код на входах блока расчета метрик	Функциональная операция
0	An-1 ..... A2 A1 0	0 An-1 ..... A2 A1 0/1	Чтение предыдущих метрик и треков из ячейки RAM узла четного источника
1	An-1 ..... A2 A1 1	0 An-1 ..... A2 A1 0/1	Чтение предыдущих метрик и треков из ячейки RAM узла нечетного источника
2	0 An-1 ..... A2 A1	0 An-1 ..... A2 A1 0/1	Запись выживших метрик и треков в ячейку RAM узла младшего приемника
3	1 An-1 ..... A2 A1	1 An-1 ..... A2 A1 0/1	Запись выживших метрик и треков в ячейку RAM узла старшего приемника

Алгоритм выполняется за четыре такта. На каждом такте битовый вектор на выходе генератора номеров узлов An An-1 An-2 ..... A2 A1 A0, трансформируется мультиплексором адресов в код на входах адресных шин RAM метрик и треков, как указано в таблице. Разрядность шин адресов равна предполагаемому максимально допустимому кодовому ограничению CLmax = n, т. е. на единицу меньше разрядности генератора узлов.

Аналогичным образом, вектор An An-1 An-2 ..... A2 A1 поступает на входы схемы определения метрик ребер блока обработки метрик. Младший бит игнорируется, так как схема содержит аппаратные средства параллельного подсчета метрик ребер для четного и нечетного адресов (условно показано, как 0/1).

На рис. 3 показана схема управления кодами и адресами декодера Витерби для случая CLmax = 15. При настройке декодера на работу с меньшими значениями CL старший бит на входе блока обработки метрик зафиксирован в крайнем левом положении, что позволило отказаться от дополнительного мультиплексора. При этом коррекция кодов образующих полиномов осуществляется программно при их загрузке в соответствующие регистры.

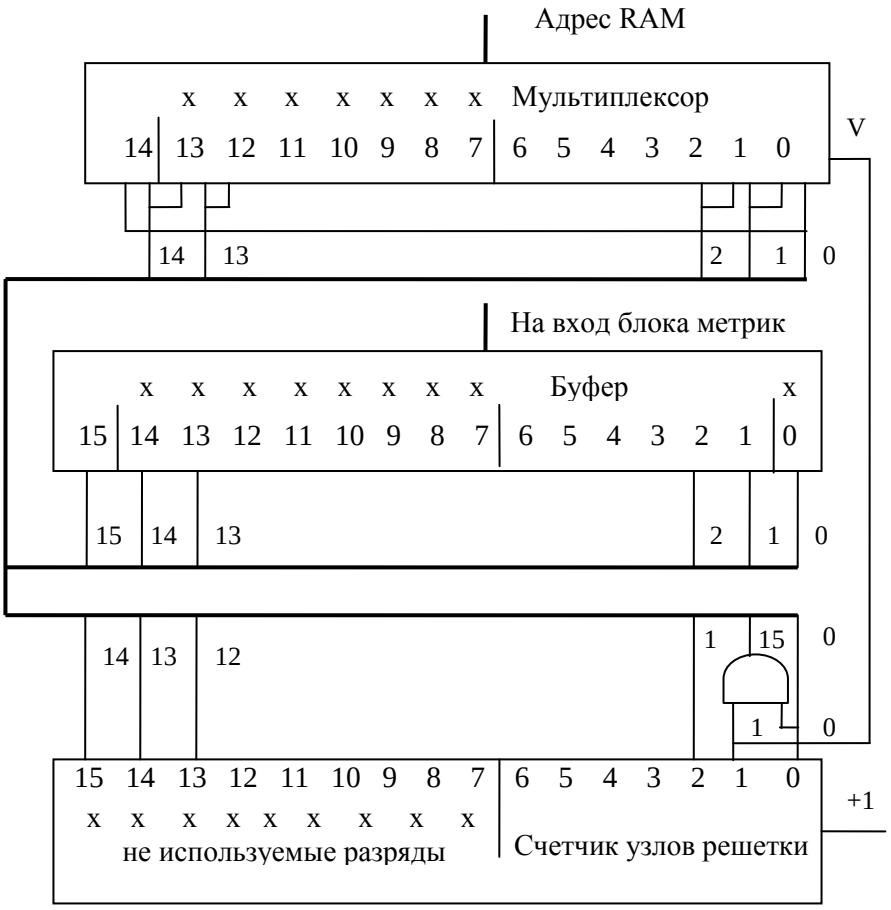

РИС. 3. Схема управления кодами и адресами декодера Витерби

Функциональная схема четырехтактного декодера Витерби применительно к библиотечным элементам и IP-ядрам ПЛИС Spartan 3 фирмы Xilinx показана на рис. 4.

Помимо упоминаемых ранее узлов на ней представлены контроллер синхронизации и связи с хост-машиной CNTR, а также регистры для хранения полиномов RgPln0, RgPln1, RgPln2, пирамидальные сумматоры по mod2, сумматоры метрик ребер Add, метрик треков Add Metr0, Add Metr1 и компаратор Comp блока обработки метрик.

Счетчик узлов CNT NODE изменяет свое состояние по переднему фронту CLK. Сначала, последовательно в течении первых двух тактов на входах адреса нулевого и первого портов формируются адреса AdrSrc0 и AdrSrc1 двух источников предыдущих метрик и треков. Прочитанные по заднему фронту CLK значе-

ния сохраняются на выходах обоих портов соответствующих блоков памяти Ram Metr и Ram Track.

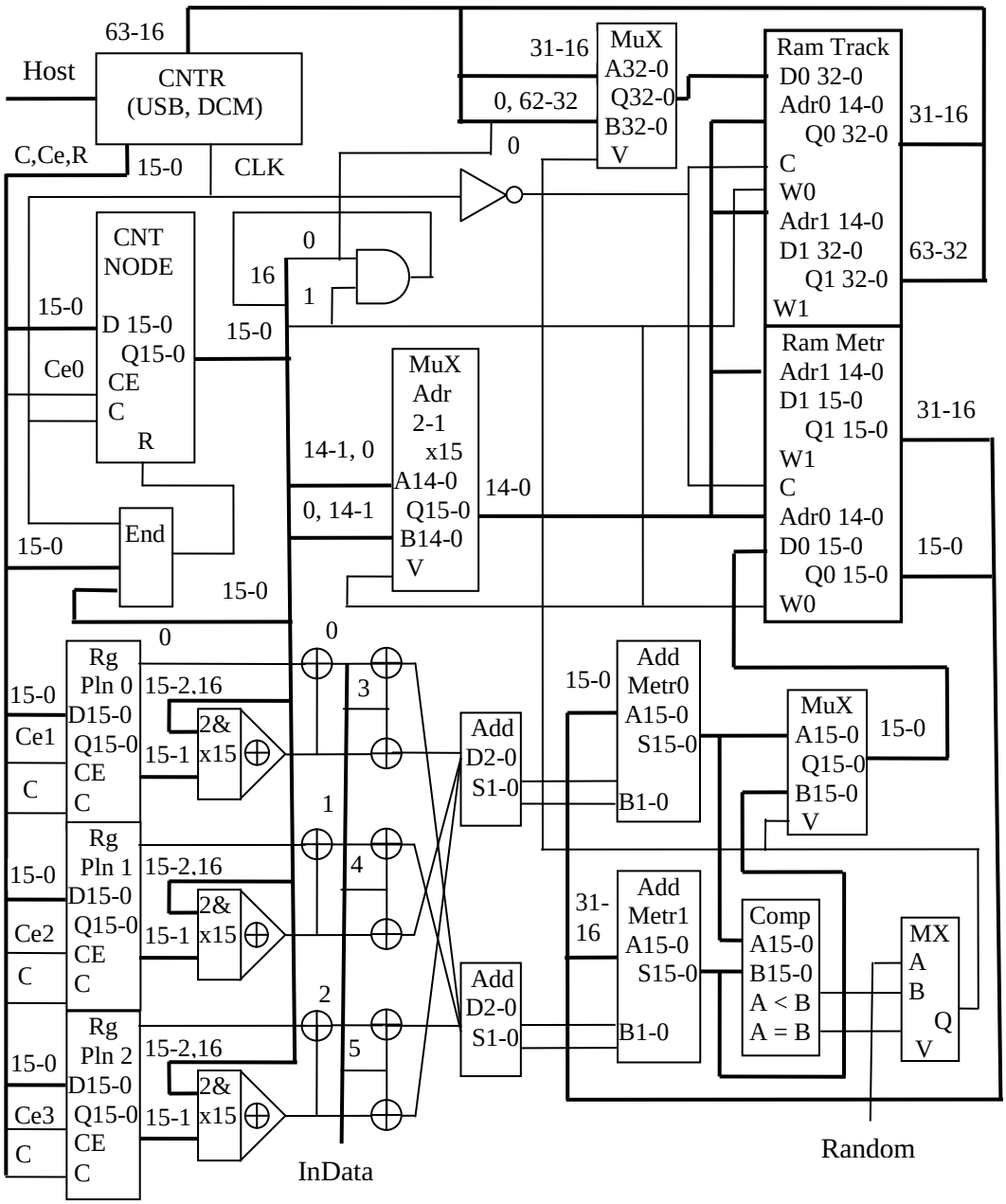

РИС. 4. Функциональная схема декодера Витерби

Формируемые комбинационной схемой блока обработки метрик на входах нулевых портов памяти новые значения метрик и треков записываются по заднему фронту CLK в последних двух тактах обработки "бабочки" Витерби.

Возможен также вариант двухтактной реализации декодера за счет одновременного чтения предыдущих метрик и треков из четных и нечетных адресов источников и записи новых метрик и треков одновременно в ячейки памяти младшего и старшего приемников. Двухпортовая память ПЛИС Xilinx это позволяет, однако при этом требуется удвоение аппаратурных ресурсов, начиная от выходных каскадов пирамидальных сумматоров по mod2 и заканчивая мультиплексорами на входах данных вторых портов RAM Track и RAM Metr. Без учета блочной памяти это несколько больше половины остальных затрат на площадь кристалла.

Небольшого снижения аппаратурных затрат при заметно больших временных потерях по сравнению с четырехтактной реализацией можно добиться за счет последовательных тактов вычисления каждой из сумм метрик перед их сравнением, с одновременным введением буферного регистра или накапливающего сумматора для хранения первой суммы.

И, наконец, использование части разрядов памяти треков для хранения двух- или трехбитовых векторов ребер позволяет убрать регистры полиномов и пирамидальные сумматоры по mod2.

Экономическая целесообразность приведенных вариантов может быть определена при известных требованиях к производительности и стоимости всего проекта в целом с учетом реальной плотности заполнения кристалла ПЛИС, в том числе другими блоками и узлами, не относящимися к декодеру Витерби.

1. *Казаринов Л.С., Попова О.В., Барбасова Т.А.* Автоматизированные информационно-управляющие системы. ч.1. – Челябинск: Изд. ЮУрГУ, 2007. – 151 с.
2. *Скляр Б.* Цифровая связь. Теоретические основы и практическое применение, 2-е изд.: Пер. с анг. – М.: Издательский дом "Вильямс", 2003. – 1104 с.
3. *Hema S., Suresh Babu V., Ramesh P.* FPGA Implementation of Viterbi – Decoder Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Corfu Island, Greece, February 16-19, 2007. – P. 162 – 167.
4. *All Spartan-6* FPGA Documentation // Available at http://www.xilinx.com.
5. *Опанасенко В.М., Лісовий О.М.* Формалізація процесу проектування обчислювальних пристроїв та систем на базі ПЛІС // Комп'ютерні засоби, мережі та системи. – К.: Ін-т кібернетики імені В.М. Глушкова НАН України. – 2009. – № 8. – С. 58 – 63.

Получено 02.10.2015