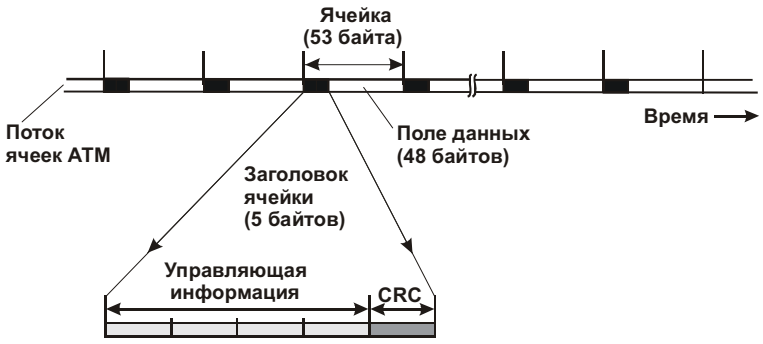

Рис. 7.23. Непосредственная передача ячеек ATM по линии связи

Ячейка состоит из заголовка и поля данных (соответственно 5 и 48 байтов). Заголовок содержит четыре байта для размещения управляющей информации (28 битов – номер логического канала плюс четыре флаговых бита) и один контрольный байт CRC (Cyclical Redundancy Check – циклический избыточный код).

Байт CRC вычисляется передатчиком и представляет собой контрольную сумму четырех предыдущих байтов. Приемник после получения ячейки проводит аналогичные вычисления и сопоставляет свой результат с имеющимся в ячейке кодом CRC. В отсутствие ошибок коды совпадают. Но как приемник определяет границы ячейки, ведь она не содержит флагов начала или конца?

7.4.2. Использование кода CRC в процессе распознавания границ ячеек

В общем виде методика поиска границ ячеек состоит в следующем. На начальном этапе поиска приемник просматривает поступающий поток ячеек (см. рис. 7.23) сквозь “непрозрачную маску с вырезанной в ней щелью”, сквозь которую видны 40 битов данных. Другими словами, перед приемником проходит “бегущая строка” длиной 5 байтов. В каждом такте строка смещается на один бит.

Приемник в каждом такте проверяет содержимое бегущей строки на предмет обнаружения в ней заголовка ячейки. Если соответствующий байт (самый новый в бегущей строке) представляет собой контрольную сумму (CRC) полученных ранее четырех байтов, то делается предположение о том, что заголовок найден. Если это предположение оправдано, то в дальнейшем моменты обнаружения заголовков будут следовать с шагом, равным длине ячейки. Если совпадение было случайным (его вероятность равна 1/256), то оно не найдет подтверждения при последующей проверке событий на периодичность.

Как следует из рис. 7.23, истинный заголовок в наихудшем случае попадет в поле зрения приемника через $53 \times 8 - 1 = 423$ такта после начала поиска. Это соответствует ситуации, при которой поиск начат как раз в тот момент, когда предыдущий заголовок только что сместился на один бит и уже не полностью попадает в область бегущей строки.

После выделения цепи правильных совпадений на фоне случайных приемник точно знает положение границ ячеек. Синхронизация с передатчиком достигнута. Но описанный анализ потока битов продолжается, и каждый новый заголовок проверяется на правильность. Если в заголовке обнаружена корректируемая ошибка, то она исправляется (код CRC позволяет вычислить положение искаженного бита). Если ошибка некорректируемая, то ячейка уничтожается. Последовательность заголовков с некорректируемыми ошибками соответствует потере синхронизации, поэтому приемник возвращается к начальному этапу поиска границ ячеек.

Далее будут рассмотрены некоторые подробности аппаратного воплощения данной методики поиска.

7.4.3. Схема “деления” двоичных чисел

Код CRC удобно называть контрольной суммой, характеризующей некоторый массив данных. Если говорить более точно, то это – контрольный остаток от “деления” некоторого двоичного числа, построенного на основе массива, на двоичное число, выбранное в качестве делителя. Слово “деления” не напрасно взято в кавычки, так как соответствующая операция, как будет показано, выполняется довольно своеобразно (“лесенкой”, но вместо вычитания используется поразрядное логическое суммирование по модулю два). Результат этой операции в общем случае не совпадает с результатом обычного деления.

Применительно к ячейкам ATM делитель представлен двоичным числом 100000111. Это число соответствует так называемому образующему полиному $X^8 + X^2 + X + 1$ (четыре его члена можно сопоставить с четырьмя единичными битами делителя). Полином, в свою очередь, имеет прямое отношение к структуре обратных связей в схеме “деления”, приведенной на рис. 7.24.

Схема содержит кольцевой сдвиговый регистр из восьми D-триггеров T и три логических элемента Исключающее ИЛИ. На вход синхронизации поступает последовательность тактовых импульсов. В каждом такте положительный фронт импульса подтверждает истинность очередного бита данных D на входе регистра.

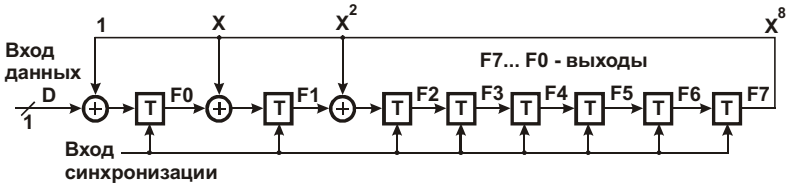

Рис. 7.24. Схема формирования (или проверки) контрольной суммы CRC

Состояние F7(n + 1) . . . F0(n + 1) регистра в такте n + 1 определяется его состоянием F7(n) . . . F0(n) в предыдущем такте n, а также новым значением входного бита данных D(n+1) и может быть выражено соотношениями (7.1):

$$
\begin{aligned}
F7(n+1) &= F6(n) \\
F6(n+1) &= F5(n) \\
F5(n+1) &= F4(n) \\
F4(n+1) &= F3(n) \\
F3(n+1) &= F2(n) \\
F2(n+1) &= F1(n) \oplus F7(n) \\
F1(n+1) &= F0(n) \oplus F7(n) \\
F0(n+1) &= D(n+1) \oplus F7(n)
\end{aligned}
\qquad (7.1)
$$

Чтобы установить взаимосвязь процесса “деления” с последовательностью состояний схемы (см. рис. 7.24), рассмотрим примеры формирования кода CRC и его использования для проверки правильности принятых данных.

7.4.4. Формирование заголовка ячейки передатчиком

На начальном этапе формирования кода CRC передатчик дополняет исходные четыре байта заголовка ячейки пятым байтом 000...0, как показано на рис. 7.25. Это эквивалентно умножению исходного 32-разрядного числа на 2^8. Полученное 40-разрядное число используется в качестве делимого; делитель равен 100000111.

Процесс преобразования 40-разрядного числа, как отмечалось, напоминает обычное деление и так же может выполняться “лесенкой”. Нули в старших разрядах делимого игнорируются. Копия делителя “пододвигается” под группу разрядов делимого, содержащую в старшем разряде лог. 1, независимо от значений последующих разрядов. Полученная пара кодов поразрядно суммируется по модулю два ($0 \oplus 0 = 0$; $0 \oplus 1 = 1$; $1 \oplus 0 = 1$; $1 \oplus 1 = 0$).

Результат дополняется справа новыми битами делимого, которые для этого “сносятся” вниз таким образом, чтобы общее число битов нового фрагмента, подлежащего обработке, соответствовало разрядности делителя. Появление нескольких нулей в старших разрядах текущего результата, как и при обычном делении, приводит к ускоренному сдвигу очередной ступени “лесенки” вправо.

Целая часть частного не используется, остаток представляет собой байт CRC. Он “прикладывается” передатчиком к четырем исходным байтам (взамен ранее “приложенного” байта 000...0) и пересылается в линию. Первым передается старший разряд байта 1, последним – младший разряд байта CRC. В таком же порядке эта группа из пяти байтов (заголовок ячейки) поступает в приемник.

Чтобы сопоставить описанный процесс получения кода CRC с процессом “прокрутки” кода через кольцевой сдвиговый регистр (см. рис. 7.24), рассмотрим ряд состояний этого регистра (рис. 7.26, левая часть).

В исходном состоянии в регистре записан нулевой код. В первом такте ($n = 1$) по положительному фронту синхроимпульса в левый разряд регистра принимается старший бит 40-разрядного числа (делимого). Поскольку в нашем примере этот бит равен нулю, состояние регистра не изменяется. Во втором, третьем и четвертом тактах при вводе последующих нулевых битов делимого состояние регистра остается нулевым. В пятом такте старший значащий разряд (лог. 1) делимого заносится в регистр, и в последующих тактах наблюдается продвижение группы единиц в направлении выхода F7.

В двенадцатом такте лидирующая лог. 1 достигает выхода F7. Поэтому логические элементы Исключающее ИЛИ во входных каскадах регистра настраиваются на режим инвертирования соответствующих сигналов, распространяющихся слева направо. Инвертирование “проявляется” в тринадцатом такте, в результате в регистре фиксируется код 01001111. В этом коде старший разряд находится справа, что создает некоторые неудобства при его рассмотрении. Перепишем код в нормальном виде (старший разряд – слева): 11110010.

Этот код совпадает с полученным на первой ступени “лесенки деления”, а именно, с кодом, расположенным под первой горизонтальной чертой этой ступени. Продолжая сопоставление, можно заметить полное соответствие набора состояний регистра правилам “деления лесенкой”. Так, число ступеней равно одиннадцати. Столько же раз логические элементы Исключающее ИЛИ во входных каскадах регистра настраиваются на режим инвертирования соответствующих сигналов, распространяющихся слева направо. Столько же раз сигнал F7 равен лог. 1 (этот сигнал отображает последовательный код целой части частного, которая отбрасывается).

В сороковом такте ($n = 40$) в регистре фиксируется код CRC.

7.4.5. Проверка правильности заголовка ячейки приемником

Итак, рассматривается ситуация, при которой в приемник поступают пять байтов, из которых пятый – код CRC, построенный на основе обработки первого – четвертого байтов. Приемник решает такую задачу: действительно ли пятый байт представляет собой правильный код CRC, т. е. соответствует ли он контрольной сумме предшествующих четырех байтов?

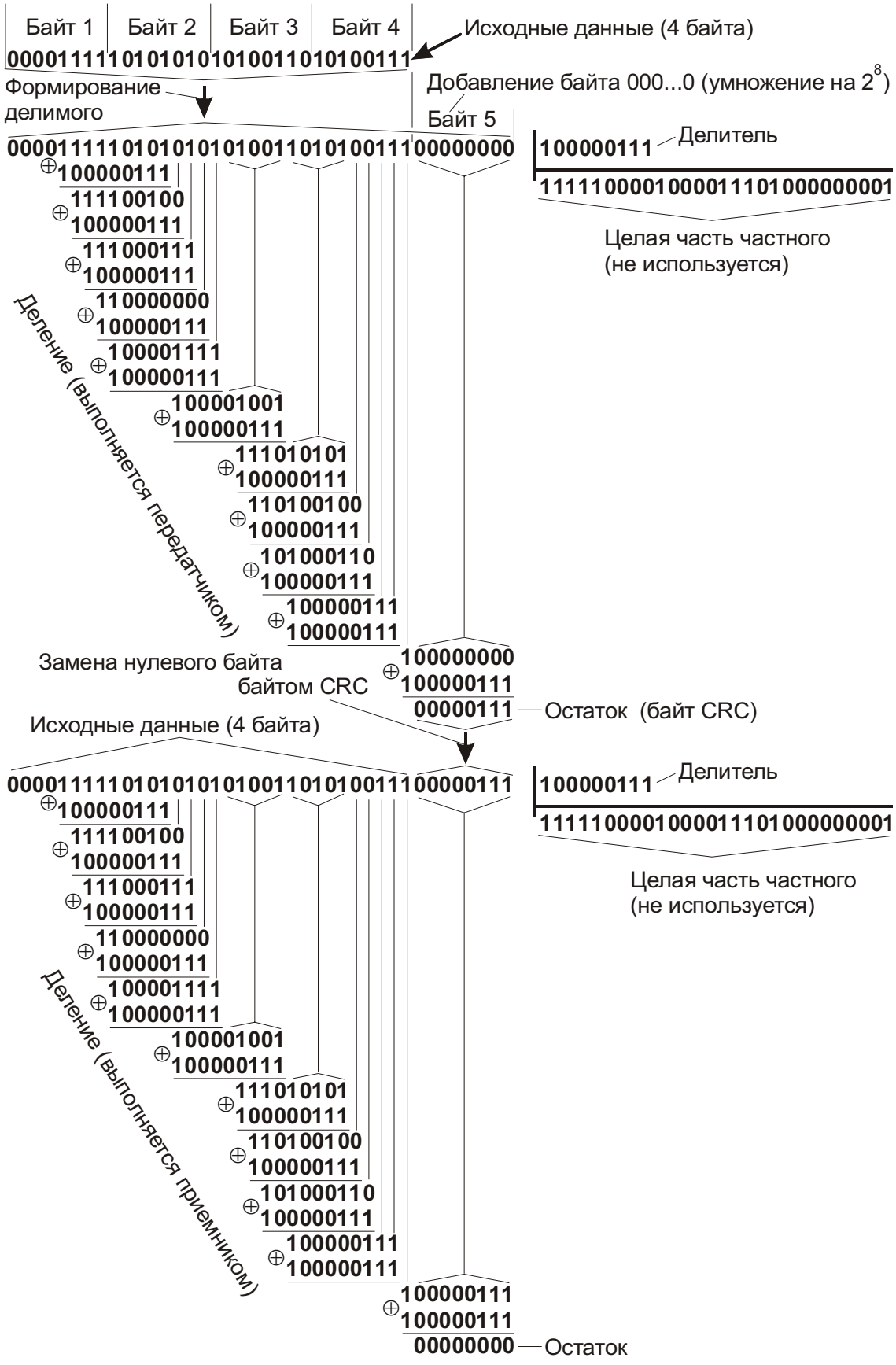

Рис. 7.25. Формирование кода CRC передатчиком (верхняя часть рисунка) и проверка полученных данных приемником (нижняя часть)

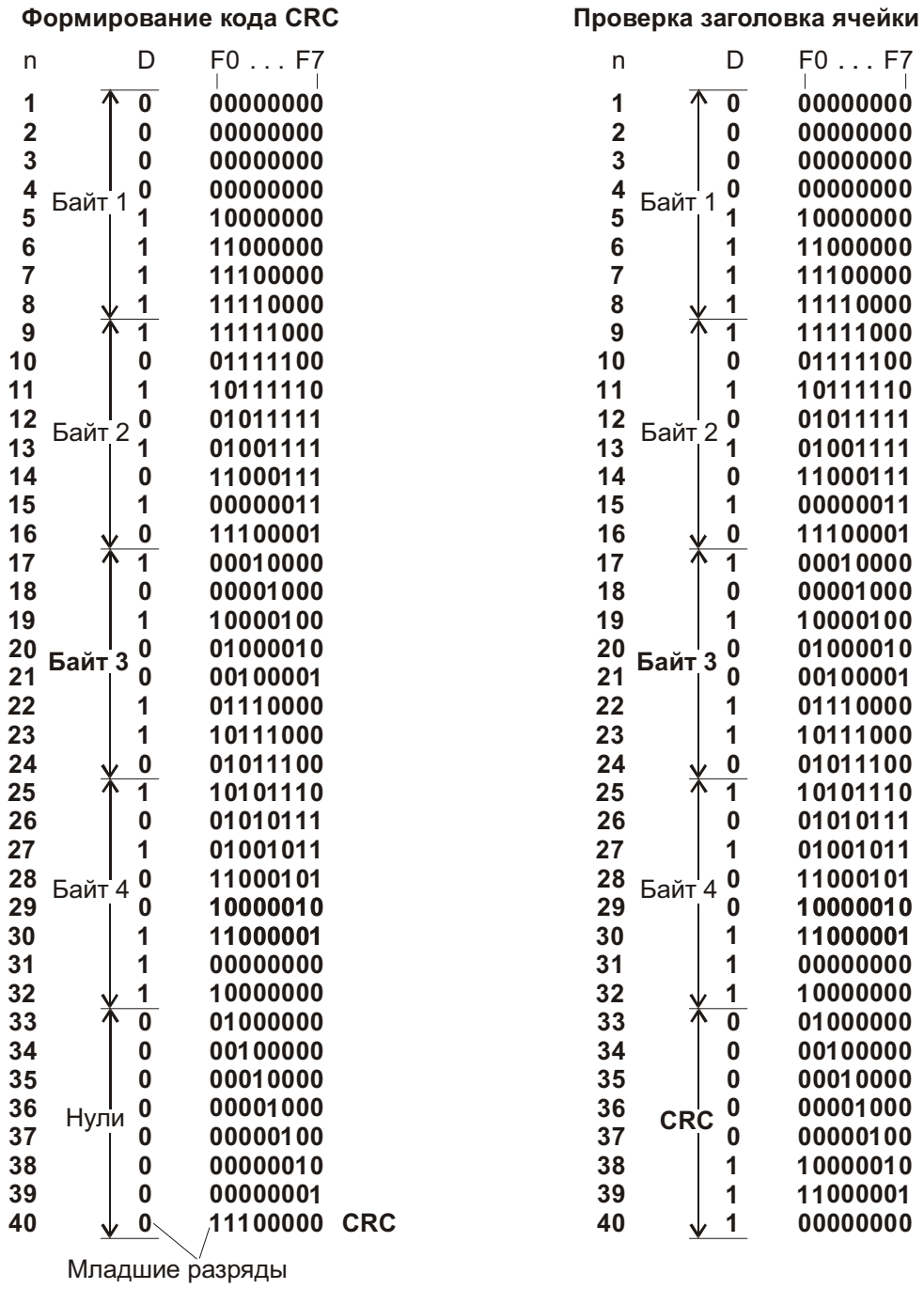

Рис. 7.26. Формирование кода CRC передатчиком (левая часть рисунка) и проверка заголовка ячейки приемником (правая часть) с использованием кольцевого сдвигового регистра, показанного на рис. 7.24; n – номер такта

Вернемся к рис. 7.25 и рассмотрим его нижнюю часть. Делимое содержит в младшем байте код 00000111, который соответствует коду CRC, вычисленному передатчиком. Деление “лесенкой” на десяти ступенях дает те же результаты, что и ранее, но на одиннадцатой ступени получаем нулевой остаток. Это свидетельствует о том, что контрольная сумма правильная (что и требовалось проверить).

В правой части рис. 7.26 показан ряд состояний кольцевого сдвигового регистра (см. рис. 7.24) приемника при проверке правильности кода CRC. Отличия от группы состояний аналогичного регистра передатчика наблюдаются в тактах 38 – 40. В част-

ности, в контрольном, сороковом такте имеем нулевое содержимое регистра, что подтверждает правильность приема заголовка.

Отметим, что нулевые коды получены также в тактах 1 – 4, 31; но в нашем примере заранее известно, что эти такты промежуточные, поэтому моменты получения промежуточных нулевых кодов не являются для нас значимыми событиями. Иное дело – в “реальной жизни”, когда такие события не должны оставаться незамеченными.

7.4.6. Поиск заголовка в непрерывном битовом потоке данных

Рассмотренный ранее способ проверки правильности кода CRC с помощью схемы, приведенной на рис. 7.24, применим лишь в идеальной “статической” ситуации, когда заранее известно, что начальное состояние сдвигового регистра – нулевое, а в первом такте (n = 1) в регистр заносится старший бит заголовка.

Реальная ситуация динамична, в том смысле, что в поле зрения приемника находится бегущая строка из 40 битов. В каждом такте содержимое строки смещается на один бит. Поэтому в каждом такте нужно успеть проверить правильность пятого байта – предположительно кода CRC. Если применить рассмотренную ранее методику, то нужно в начале каждого такта установить кольцевой сдвиговый регистр в нуль, а оставшийся тактовый интервал разбить на 40 микротактов для выполнения процедуры, показанной в правой части рис. 7.26. Но такое решение слишком “прямолинейно”, чтобы быть эффективным.

Рассмотрим решение, предложенное в [23] (рис. 7.27). Здесь проверка очередного 40-разрядного кода выполняется за один такт в конвейерном режиме. Новый бит, вводимый в бегущую строку, обычным образом учитывается в новой контрольной сумме; но одновременно с этим новая контрольная сумма корректируется из-за того, что один бит “вытолкнут” за пределы бегущей строки, и схема должна уничтожить его былой вклад в контрольную сумму. Теперь всё по порядку.

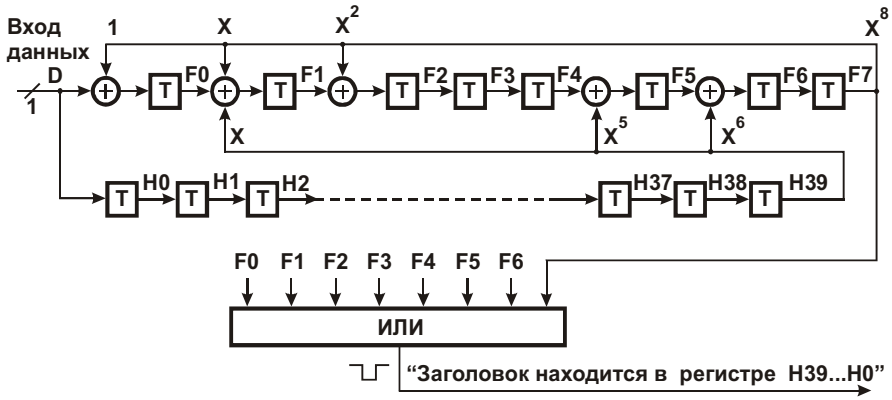

рис. 7.27. Схема распознавания ячейки в непрерывном потоке битов. Цепи синхронизации триггеров не показаны

Сравнивая схемы, показанные на рис. 7.24 и рис. 7.27, можно заметить, что последняя представляет собой расширение первой. Кольцевой сдвиговый регистр содержит дополнительные цепи коррекции результата. Введен 40-разрядный сдвиговый регистр для хранения бегущей строки, в которой ожидается появление заголовка ячейки (точнее, кода с правильной контрольной суммой, претендующего на то, чтобы именоваться заголовком). Элемент ИЛИ регистрирует появление нулевого кода в кольцевом сдвиговом регистре. Все D-триггеры T синхронизируются общим тактовым сигналом.

Состояние F7(n + 1) . . . F0(n + 1) кольцевого сдвигового регистра в такте n + 1 определяется состоянием F7(n) . . . F0(n) этого регистра в предыдущем такте n, со-

стоянием разряда H39(n) в такте n, а также новым значением входного бита данных D(n+1) и может быть выражено следующими соотношениями:

$$
\begin{aligned}
F7(n+1) &= F6(n) \\
F6(n+1) &= F5(n) \oplus H39(n) \\
F5(n+1) &= F4(n) \oplus H39(n) \\
F4(n+1) &= F3(n) \\
F3(n+1) &= F2(n) \\
F2(n+1) &= F1(n) \oplus F7(n) \\
F1(n+1) &= F0(n) \oplus F7(n) \oplus H39(n) \\
F0(n+1) &= D(n+1) \oplus F7(n)
\end{aligned}
\qquad (7.2)
$$

Процесс распознавания заголовка ячейки поясняется рис. 7.28.

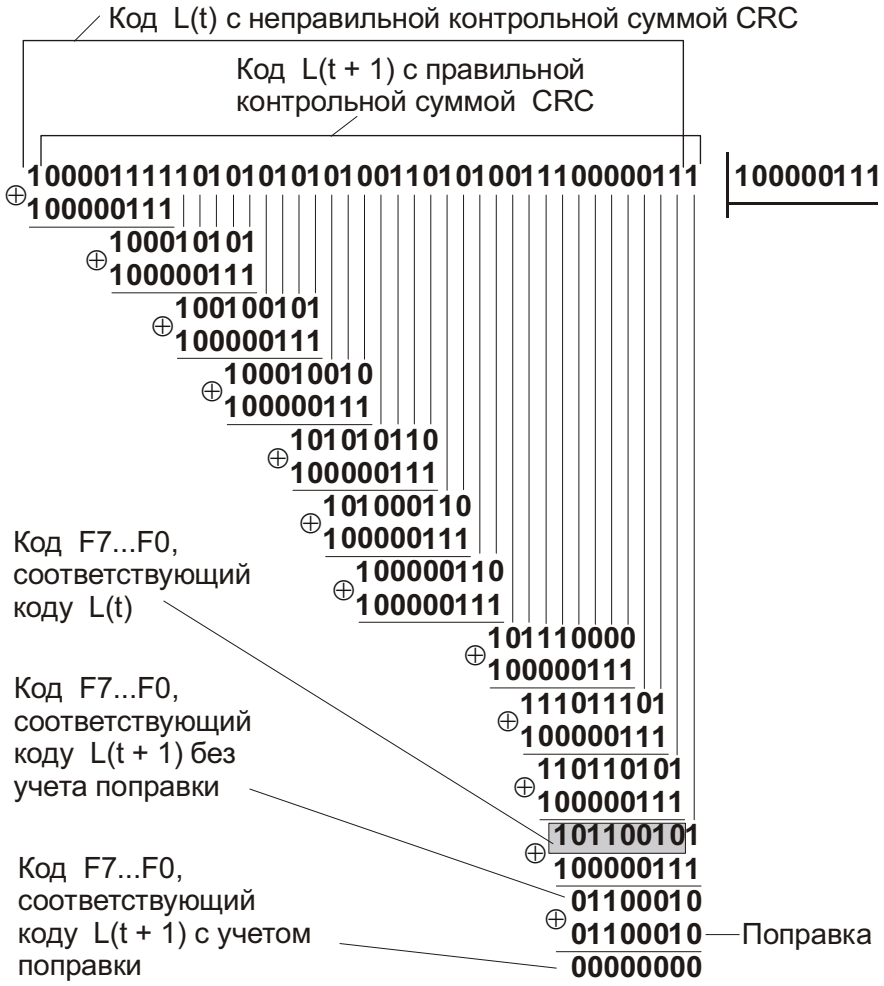

рис. 7.28. Конвейерная обработка бегущей строки на примере анализа перекрывающихся кодов L(t) и L(t + 1) с неправильной и правильной контрольными суммами

Предположим, что в некоторый момент t бегущая строка (регистр H39 – H0) содержит код L(t). Примем пока “на веру”, что в этот же момент в кольцевом сдвиговом регистре присутствует результат 40-тактной проверки кода L(t) по приведенной ранее методике. Этот результат (код 10110010, выделенный на рисунке серым фоном) отличен от нулевого, что находится в согласии с тем, что пятый байт кода L(t) не является контрольной суммой (CRC) четырех предыдущих байтов.

В момент t + 1 в бегущей строке фиксируется код L(t + 1). Этот код мы уже рассматривали в предыдущем примере. Он содержит правильную контрольную сумму

(CRC), равную 00000111. Из рисунка следует, что простой ввод в кольцевой сдвиговый регистр очередного бита, т. е. простой (без поправки) переход от кода L(t) к коду L(t + 1) приводит к результату, равному 01100010. Этот результат нельзя считать правильным, так как при правильной контрольной сумме результат должен быть нулевым. Таким образом, если не учитывать поправку, то операцию можно выполнить быстро, за один такт, но, к сожалению, неправильно.

В чем же тут дело? Проблема состоит в том, что результат, равный 01100010, получен при обработке не 40-, а 41-разрядного числа, в котором старший разряд принадлежит коду L(t), а остальные – коду L(t + 1). Иными словами, в полученном результате неоправданно учитывается вклад в контрольную сумму исчезнувшего бита, отброшенного за пределы бегущей строки. Нужно уничтожить этот вклад. Но каков он?

Чтобы ответить на этот вопрос, отметим, что отброшенный разряд отображает число 2^{40} = 1000...0 (сорок нулей после единицы). Если “разделить” (например, “лесенкой”) это число на наш делитель (100000111), то получим в остатке искомый вклад исчезнувшего бита. Он равен 01100010 (соответствующий образующий полином: $X^6 + X^5 + X$). Остается вычесть этот мешающий вклад (поправку) из результата обработки 41-разрядного числа. Как отмечалось, “деление” выполняется довольно своеобразно, и операция вычитания заменяется поразрядным суммированием чисел по модулю два.

Такое суммирование выполняется на заключительном этапе процедуры, показанной на рис. 7.28. Конечный результат совпадает с ожидаемым (нулевым). Как следует из схемы, приведенной на рис. 7.27, введение поправки совмещено во времени с вводом очередного бита данных D, поэтому проверка кода в бегущей строке выполняется за один такт. Если бит, покинувший бегущую строку, нулевой, то поправка не нужна.

Таким образом, если в такте j проверка содержимого бегущей строки выполнена правильно, то и в следующем такте j + 1 гарантируется правильность результата (ошибками передачи пренебрегаем). Отсюда можно заключить, что схема (рис. 7.27) обеспечивает непрерывное слежение за содержимым бегущей строки, причем каждый 40-разрядный код анализируется за один такт. Конечно, для начального заполнения “конвейера” нужно потратить 40 тактов после предварительной установки всех триггеров T в нулевое состояние.

После установления надежной синхронизации приемника с передатчиком (на уровне установления границ ячеек) приемник использует код CRC для коррекции одиночных ошибок в заголовке. Это возможно, в частности, благодаря тому, что к моменту обнаружения ошибки заголовок хранится в регистре H39 – H0, поэтому можно проинвертировать ошибочный бит, т. е. исправить его (цепи коррекции одиночных ошибок в заголовке ячейки на рисунке не показаны).

7.4.7. Поиск заголовка в непрерывном байтовом потоке данных

Размещение ячейки внутри кадра

Ячейки ATM могут передаваться либо непосредственно по линии связи, как было показано на рис. 7.23, либо в составе более крупных структурных единиц, например кадров SDH (рис. 7.29). (SDH - Synchronous Digital Hierarchy – синхронная цифровая иерархия, европейский стандарт на средства передачи данных.)

Здесь существенно то, что аппаратура обработки кадра распознаёт его начало и преобразует поток битов в поток байтов. Это позволяет снизить необходимую скорость обработки данных в восемь раз. Кадр построен так, что заранее неизвестно, где размещен блок ячеек ATM. Поэтому задача отыскания границ ячеек остается актуальной и при байтовом потоке данных.

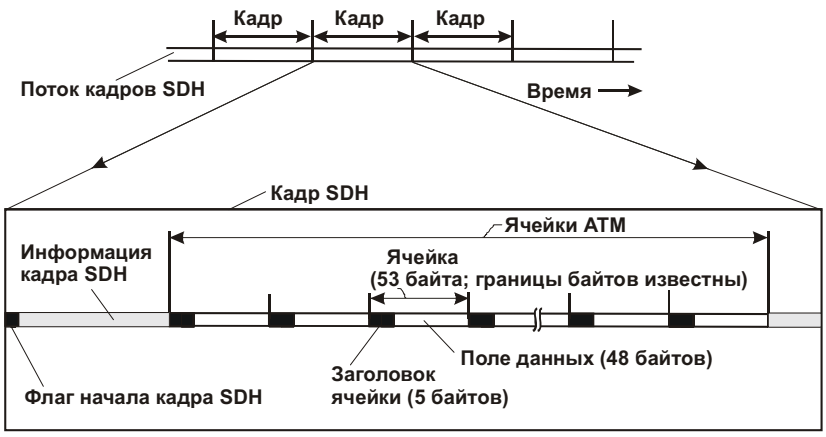

Рис. 7.29. Передача ячеек ATM по линии связи в упакованном виде. Ячейки помещены в кадры SDH

Схема распознавания заголовка ячейки в байтовом потоке данных

Решение, приведенное на рис. 7.30, представляет собой “распараллеленный” аналог рассмотренной ранее схемы (см. рис. 7.27).

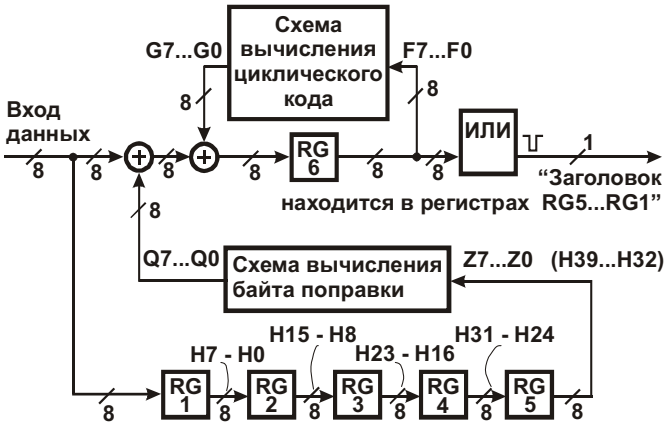

Рис. 7.30. Схема распознавания заголовка ячейки в непрерывном потоке данных – байтовый вариант. Общая цепь синхронизации регистров RG1 – RG6 не показана

Группа параллельных регистров RG5 – RG1 представляет собой байтовый сдвиговый регистр для хранения содержимого бегущей строки. Регистр RG6 хранит текущий результат вычисления циклического кода. Схема содержит две последовательно включенные группы логических элементов Исключающее ИЛИ для вычисления циклического кода с учетом поправки, аналогично тому, как это делалось в предыдущей схеме, но в байтовом воплощении. Восьмивходовый логический элемент ИЛИ регистрирует появление нулевого кода в регистре RG6.

Структура других составных частей этой схемы приведена на рис. 7.31.

Схемы вычисления циклического кода и байта поправки содержат по восемь элементов Исключающее ИЛИ. Конфигурация связей в этих схемах определяется приведенными далее логическими выражениями, которые мы получим на основе рассмотренных ранее соотношений (7.2).

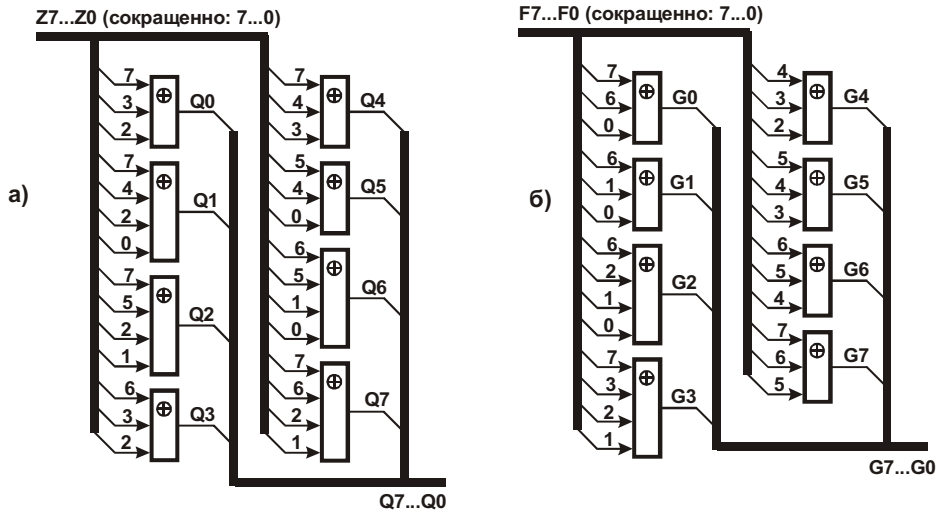

Рис. 7.31. Составные части схемы, приведенной на рис. 7.30: а – схема вычисления байта поправки; б – схема вычисления циклического кода

Логические соотношения для перехода от битового потока данных к байтовому

Напомним, что соотношения (7.2) описывают процесс вычисления циклического кода при последовательной обработке данных схемой, приведенной на рис. 7.27. Такт n в соотношениях (7.2) соответствует предыдущему состоянию схемы, такт n + 1 отображает текущее состояние. В этих же терминах предыдущее состояние “байтовой” схемы (рис. 7.30) характеризуется следующим набором переменных:

F7(n), F6(n), F5(n), F4(n), F3(n), F2(n), F1(n), F0(n);
Z7(n), Z6(n), Z5(n), Z4(n), Z3(n), Z2(n), Z1(n), Z0(n).

Текущее состояние входного байта данных D простирается от “настоящего времени” (n + 8) на семь тактов в “прошедшее”:

D(n + 8), D(n + 7), D(n + 6), D(n + 5), D(n + 4), D(n + 3), D(n + 2), D(n + 1).

Текущее состояние регистра RG6 характеризуется следующим набором переменных:

F7(n + 8), F6(n + 8), F5(n + 8), F4(n + 8), F3(n + 8), F2(n + 8), F1(n + 8), F0(n + 8).

Установим зависимость текущего состояния регистра RG6 от текущего состояния входного байта данных D и предыдущего состояния схемы, принимая во внимание следующие соотношения:

$$Z7(n) = H39(n),\ Z6(n) = H38(n),\ Z5(n) = H37(n),\ \ldots,\ Z0(n) = H32(n)$$

$$H39(n) = H38(n-1) = H37(n-2) = H36(n-3) = H35(n-4) = \ldots$$

Первая цепь равенств принята по определению, вторая описывает работу битового сдвигового регистра.

Запишем первое соотношение из (7.2) в следующем виде:

$$F7(n + 8) = F6(n + 7). \qquad (7.3)$$

С учетом второго соотношения из (7.2) равенство (7.3) можно продолжить:

$$F7(n + 8) = F6(n + 7) = F5(n + 6) \oplus H39(n + 6). \qquad (7.4)$$

Продолжаем проникать во все более глубокие “археологические слои”, учитывая оставшиеся соотношения (7.2). Цепь равенств (7.4) приобретает следующий вид, при котором каждая последующая строка отображает очередной, более глубокий “слой” (некоторое исключение представляет последняя строка):

$$\begin{aligned}
&F7(n + 8) = \\
&= F6(n + 7) = \\
&= F5(n + 6) \oplus H39(n + 6) = \\
&= F4(n + 5) \oplus H39(n + 5) \oplus H38(n + 5) = \\
&= F3(n + 4) \oplus H38(n + 4) \oplus H37(n + 4) = \\
&= F2(n + 3) \oplus H37(n + 3) \oplus H36(n + 3) = \\
&= F1(n + 2) \oplus F7(n + 2) \oplus H36(n + 2) \oplus H35(n + 2) = \\
&= F0(n + 1) \oplus F7(n + 1) \oplus H39(n + 1) \oplus F6(n + 1) \oplus H35(n + 1) \oplus H34(n + 1) = \\
&= D(n + 1) \oplus F7(n) \oplus F6(n) \oplus H38(n) \oplus F5(n) \oplus H39(n) \oplus H34(n) \oplus H33(n).
\end{aligned}$$

После упорядочения в последней строке последовательности “слагаемых”, получим:

$$F7(n + 8) = D(n + 1) \oplus F7(n) \oplus F6(n) \oplus F5(n) \oplus H39(n) \oplus H38(n) \oplus H34(n) \oplus H33(n).$$

Аналогичные “раскопки глубин истории” можно провести и в отношении переменных $F6(n + 8)$, $F5(n + 8)$, $F4(n + 8)$, ..., $F0(n + 8)$. В результате получим полный набор функций перехода схемы (см. рис. 7.30) от предыдущего состояния к текущему:

$$\begin{aligned}
F7(n+8) &= D(n + 1) \oplus F7(n) \oplus F6(n) \oplus F5(n) \oplus H39(n) \oplus H38(n) \oplus H34(n) \oplus H33(n) \\
F6(n+8) &= D(n + 2) \oplus F6(n) \oplus F5(n) \oplus F4(n) \oplus H38(n) \oplus H37(n) \oplus H33(n) \oplus H32(n) \\
F5(n+8) &= D(n + 3) \oplus F5(n) \oplus F4(n) \oplus F3(n) \oplus H37(n) \oplus H36(n) \oplus H32(n) \\
F4(n+8) &= D(n + 4) \oplus F4(n) \oplus F3(n) \oplus F2(n) \oplus H39(n) \oplus H36(n) \oplus H35(n) \qquad (7.5) \\
F3(n+8) &= D(n + 5) \oplus F7(n) \oplus F3(n) \oplus F2(n) \oplus F1(n) \oplus H38(n) \oplus H35(n) \oplus H34(n) \\
F2(n+8) &= D(n + 6) \oplus F6(n) \oplus F2(n) \oplus F1(n) \oplus F0(n) \oplus H39(n) \oplus H37(n) \oplus H34(n) \oplus H33(n) \\
F1(n+8) &= D(n + 7) \oplus F6(n) \oplus F1(n) \oplus F0(n) \oplus H39(n) \oplus H36(n) \oplus H34(n) \oplus H32(n) \\
F0(n+8) &= D(n + 8) \oplus F7(n) \oplus F6(n) \oplus F0(n) \oplus H39(n) \oplus H35(n) \oplus H34(n)
\end{aligned}$$

Соотношения (7.5) подтверждают правильность выбора общей структуры схемы распознавания заголовка ячейки (см. рис. 7.30). Действительно, “по большому счету” в этих соотношениях просматриваются три группы слагаемых (по модулю два):

1) байт входных данных $D(n + 8)$, $D(n + 7)$, ..., $D(n + 1)$;

2) байт, полученный на основе суммирования по модулю два определенных разрядов кода $F7(n)$, $F6(n)$, $F5(n)$, …, $F0(n)$. Этот байт формируется на выходе схемы вычисления циклического кода;

3) байт, полученный на основе суммирования по модулю два определенных разрядов кода $H39(n)$, $H38(n)$, $H37(n)$, …, $H32(n)$. Этот байт формируется на выходе схемы вычисления поправки.

Теперь можно установить соответствие между соотношениями (7.5) и схемными решениями, приведенными на рис. 7.31.

Рассмотрим, например, цепь формирования старшего разряда схемы вычисления байта поправки (см. рис. 7.31, а, правый нижний логический элемент Исключающее ИЛИ). На входы этого элемента поступают сигналы Z7, Z6, Z2, Z1 или, что то же самое, H39, H38, H34, H33. Это согласуется с группой соответствующих переменных в первой строке соотношений (7.5).

Цепь формирования младшего разряда схемы вычисления циклического кода (см. рис. 7.31, *б*) построена на основе трехвходового логического элемента Исключающее ИЛИ, размещенного слева вверху. На входы этого элемента поступают сигналы F7, F6 и F0. Это согласуется с группой соответствующих переменных в последней строке строке соотношений (7.5).

7.5. Распознавание и восстановление искаженных кадров при передаче данных по радиоканалу

В системе мобильной телефонной связи (рис. 7.32) предусмотрена одновременная передача “оцифрованных” речевых и иных данных по радиоканалу. Данные, не относящиеся к передаче речи, пересылаются в паузах между словами и фразами [28]. Дальнейшее описание относится именно к таким данным.

Рис. 7.32. Фрагмент системы мобильной телефонной связи

Потоки данных состоят из последовательностей кадров. Кадр первоначально формируется в памяти передающего устройства и представляет собой группу байтов, в которой первый и два последних служебные, а остальные содержат полезные данные, объединенные в сообщение (рис. 7.33).

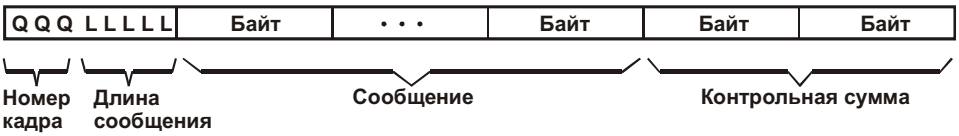

Рис. 7.33. Структура кадра до его выдачи в радиоканал

Биты Q первого байта задают порядковый номер кадра, биты L определяют длину сообщения. Если все биты L нулевые, то сообщение не содержит ни одного байта. Такой “АСК-кадр” (от Acknowledge – ответ) рассматривается как подтверждение успешного приема кадра, содержащего сообщение. (Ответный кадр пересылается в противоположном направлении по отношению к кадру, несущему сообщение.) Два последних байта представляют собой контрольную сумму (16-разрядный циклический код CRC) всех предшествующих байтов кадра.

Дальнейшая подготовка кадра к выдаче в радиоканал состоит в его дроблении на отдельные асинхронные посылки (см. п. 1.2, рис. 1.5). При этом каждый байт преобразуется в две посылки, как показано на

рис. 7.34.

Преобразование состоит в следующем. Сначала байт D1 … D8 делится на два полубайта: D1 … D4 и D5 … D8. Им присваиваются четырехразрядные порядковые номера: N1 … N4 и M1 … M4. Эти номера “склеиваются” с соответствующими полубайтами, в результате образуются два байта:

N1 … N4 D1 … D4 и M1 …M4 D5 … D8.

На заключительной стадии преобразования к обоим байтам добавляются стартовые биты ST, биты P контроля по четности (нечетности) и стоповые биты ST.

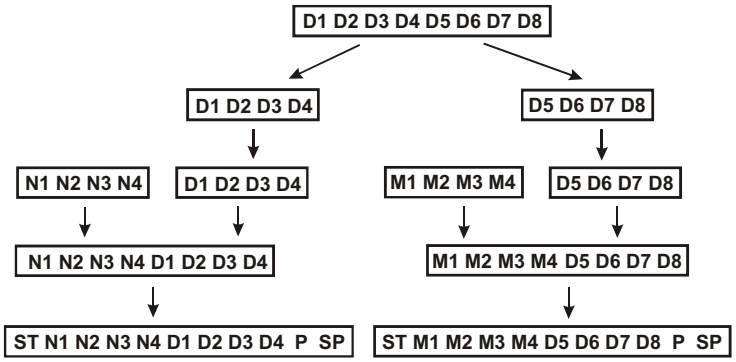

Рис. 7.34. Преобразование байта в две старт-стоповые посылки

В результате преобразования исходный кадр дробится на множество посылок (их число равно удвоенному числу байтов кадра). Они выдаются в радиоканал в паузах между “всплесками” передачи речевой информации. Поэтому поток данных неравномерен и может содержать как следующие друг за другом вплотную, так и разрозненные группы посылок или отдельные посылки. В процессе передачи по радиоканалу часть посылок может быть потеряна из-за возможных замираний сигнала или действия помех. Приемник может также получать ложные посылки, которые вклинились в неравномерный поток правильных. Чтобы обеспечить надежный обмен данными в условиях замираний сигнала и действия помех, предлагаются три простые, но примечательные идеи.

Идея 1. Проверка правильности порядка следования посылок слежением за динамикой изменения номеров полубайтов. Как было показано, каждая посылка содержит полубайт данных и предшествующий ему четырехразрядный номер этого полубайта. В отсутствие ошибок приемник должен регистрировать некоторую заранее заданную последовательность номеров полубайтов, например такую: … 0, 1, 2, 3, …, F, 0, … (использована шестнадцатиричная форма записи чисел). Если получена последовательность …0, 1, 2, 7, 3, 4, 5, 6, …, то можно сделать вывод о том, что посылка с порядковым номером 7 нарушает динамику изменения номеров и явно лишняя. Поэтому приемник просто отбрасывает ее. Аналогично в последовательности … 0, 1, 2, 4, 5, 6, … недостает цифры 3. Это означает, что соответствующая посылка потеряна по пути к приемнику. Поэтому передача ошибочного кадра повторяется в связи с неполучением подтверждения правильного приема данных.

Идея 2. Расширение диапазона нумерации посылок при ограниченной разрядности номеров. С помощью четырехразрядного двоичного кода можно отображать числа в диапазоне $0 - F_{16}$. Поэтому на первый взгляд представляется естественной простая последовательная нумерация посылок: … 0, 1, 2, …, 9, A, B, C, D, E, F, 0, 1, 2, … . Однако такая нумерация повторяется с периодом, равным 16, а кадр может быть представлен десятками и сотнями посылок. Действительно, можно построить достаточно большой кадр, если принять условие, что длина сообщения (см. код LLLLL на рис. 7.33) выражается не в отдельных байтах, а в более крупных единицах, например, в группах из четырех байтов.

Можно ли с помощью четырехразрядных кодов пронумеровать сотню или более посылок так, чтобы приемник мог уверенно определять положение любой из них относительно начала кадра? На первый взгляд, ответ на этот вопрос может быть только отрицательным. Но решение есть, и в этом мы сейчас убедимся.

Пример. Предположим, что составляющие кадр посылки пронумерованы с помощью такой последовательности номеров:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 0, 3, 6, 9, C, F, 2, 5, 8, B, E, 1, 4, 7, A, D, 0, 5, A, F, 4, 9, …, 0, 7, E, 5, C, …

Первые 16 посылок пронумерованы в обычном порядке, начиная с нулевого номера. Следующие 16 посылок также нумеруются, начиная с нуля, но номер наращивается с шагом, равным трем. Следующие 16 посылок нумеруются с нуля с шагом 5, следующие – с шагом 7, затем с шагами 9, 11, 13 и 15. В этом примере в качестве шагов выбраны восемь последовательно возрастающих чисел, не превышающих 15 и не имеющих общего делителя с числом 16, т. е. нечетные числа. Таким образом, получена *уникальная* последовательность из $16 \times 8 = 128$ *неуникальных* номеров. Эту последовательность можно продолжить, применив какие-либо заранее учтенные в протоколе обмена дополнительные правила ее формирования. Например, можно повторить последовательность, поменяв местами номера, размещенные в первых и последних пяти позициях каждой группы из 16 посылок и т. д. Иными словами, возможности построения такого рода уникальных последовательностей номеров практически безграничны.

Приемник просматривает полученную последовательность и проверяет ее правильность. Как уже было показано, зная вид последовательности, можно выявить и отбросить лишние номера и соответствующие посылки, если они есть, и зафиксировать пропущенные номера посылок. Так как последовательность уникальна, то можно отыскать ее начало и конец, т. е. идентифицировать первую и последнюю посылки кадра.

Отметим, что рассмотренная идея применима также и к нумерации кадров – наличие трех битов QQQ для указания номера кадра вовсе не означает, что соответствующий диапазон составляет только восемь номеров.

Идея 3. Усовершенствованное мажоритарное восстановление ошибочных байтов кадра. Если приемник получил все посылки, относящиеся к кадру, то он реконструирует байты в последовательности, обратной той, которая была приведена на рис. 7.34. Для этого он отбрасывает служебные биты ST, P и SP выбранной пары посылок, затем уничтожает их номера N1 – N4 и M1 – M4 и склеивает полубайты. В результате формируется первоначальный байт D1 – D8.

После формирования всех байтов кадра приемник проверяет его контрольную сумму (см. рис. 7.33). Если контрольная сумма правильная, то приемник посылает уведомление об этом источнику кадра. (Напомним, что в качестве уведомления используется кадр с нулевым кодом LLLLL длины сообщения.)

Если контрольная сумма неправильная, или в кадре отсутствуют некоторые байты, то приемник не посылает уведомление, сохраняет имеющуюся информацию о кадре и ожидает поступления копии ошибочного кадра. При поступлении копии проверяется ее контрольная сумма. Если она правильная, то посылается уведомление об этом, в противном случае приемник сопоставляет две ошибочные версии кадра и пытается собрать из них один полноценный кадр. Если это удалось сделать и контрольная сумма правильная, то посылается уведомление, иначе приемник ждет поступление новой копии и т. д.

При наличии нескольких копий кадра применяется обычный метод мажоритарного выбора, при котором используется “голосование по большинству”. Например если два одноименных байта из трех совпадают, то их значение принимается в качестве истинного. Если все три байта разные, то голосование переносится на уровень битов. Это позволяет в некоторых ситуациях, когда одиночные ошибки распределены по разным битам, восстановить правильный байт. Но если, например, из пяти одноименных байтов два совпадают между собой, но отличаются от трех других, которые, в свою очередь, не совпадают друг с другом в любых сочетаниях, то эти совпадающие байты принимаются в качестве истинных, хотя, строго говоря, они – в меньшинстве. Конечно, окончатель-

ное решение об отсутствии ошибок в кадре принимается после проверки его контрольной суммы.

7.6. Распознавание флагового кода, содержащего ошибки

В некоторых системах, например в системе мобильной связи, передача кадров часто происходит на фоне помех, которые искажают отдельные биты или группы битов. В частности, не исключена возможность появления искажений во флаговых кодах, обозначающих начало кадра. С этими искажениями нельзя не считаться, так как если ориентироваться только на получение безошибочных флаговых кодов, то из-за их нерегулярного поступления трудно установить или поддерживать ранее установленную кадровую синхронизацию. Задача распознавания флаговых кодов, в том числе, содержащих ошибки, может быть решена с помощью схемы, приведенной на рис. 7.35, *а* [34].

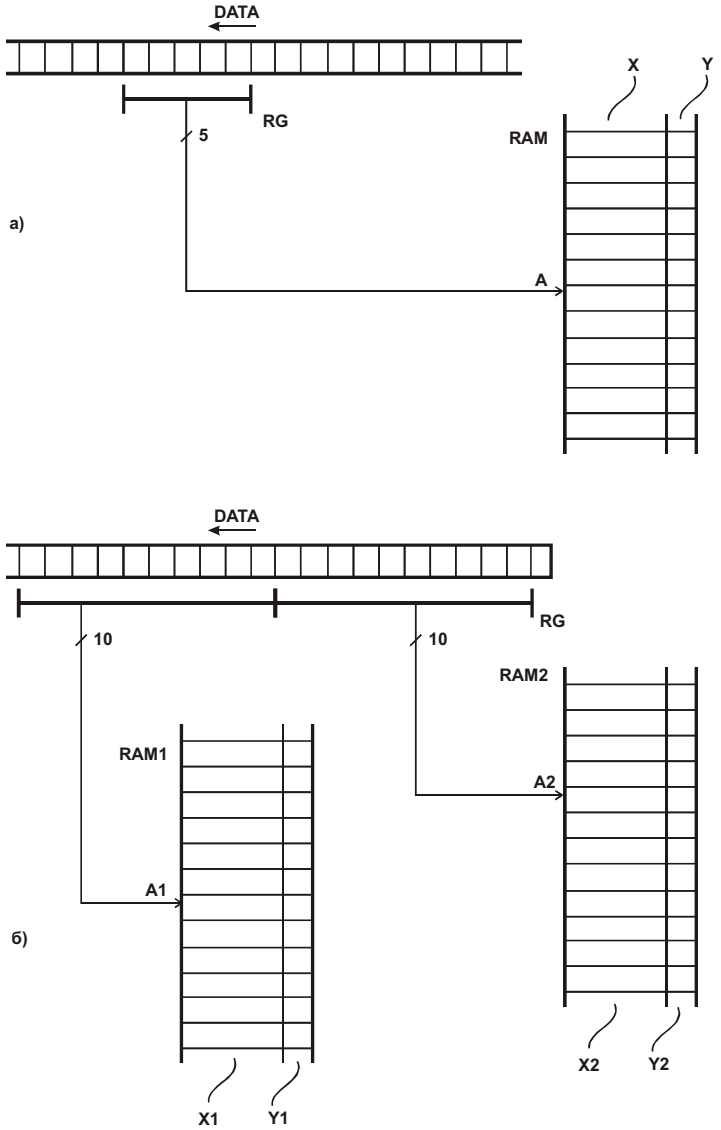

рис. 7.35. Схема поиска флагового кода: *а* – с использованием одного блока памяти; *б* – с использованием двух блоков памяти

Входной поток данных DATA проходит через сдвиговый регистр RG. Этот процесс можно интерпретировать как просмотр движущейся последовательности битов через неподвижное пятиразрядное окно (направление движения данных показано

стрелкой). Параллельный код из регистра RG поступает на адресные входы блока памяти RAM.

Предположим, что в искомом флаговом коде 10101 (малая разрядность выбрана для удобства иллюстрации идеи) допустима одиночная ошибка. Тогда помимо этого кода схема поиска должна распознавать еще пять кодов, отличающихся от искомого в одном из разрядов: 00101, 11101, 10001, 10111 и 10100. Для распознавания правильного и пяти близких к нему ошибочных кодов в память предварительно записывается информация, представленная в Таблица 7.3 табл. 7.3.

Таблица 7.3 табл. 7.3

Таблица кодировки блока памяти (см. рис. 7.35, *а*)

Адрес	Эталонный код (X)	Число ошибок (Y)
00000	-	-
00001	-	-
00010	-	-
00011	-	-
00100	-	-
00101	10101	1
00110	-	-
00111	-	-
01000	-	-
01001	-	-
01010	-	-
01011	-	-
01100	-	-
01101	-	-
01110	-	-
01111	-	-
10000	-	-
10001	10101	1
10010	-	-
10011	-	-
10100	10101	1
10101	10101	0
10110	-	-
10111	10101	1
11000	-	-
11001	-	-
11010	-	-
11011	-	-
11100	-	-
11101	10101	1
11110	-	-
11111	-	-

Примечание. Символ “-” обозначает отсутствие данных и может кодироваться отдельным битом

Из таблицы следует, что точное или приближенное (с ошибкой в одном разряде) совпадение входного кода с заданным флаговым кодом сопровождается выдачей из памяти эталонного кода X и числа ошибок Y. В данном примере эталонный код уника-

лен (10101), поэтому из таблицы можно было бы исключить второй столбец. Однако эталонных кодов может быть несколько, например когда каждый из них соответствует кадру определенного иерархического уровня. В этом случае по считанным из памяти кодам X и Y определяется ближайший эталонный код и степень близости к нему входного кода. На основе считанных из памяти данных накапливается статистическая информация, по которой можно провести правильное разбиение потока данных на кадры.

С увеличением длины флагового кода растет требуемый объем блока памяти. Так, для анализа 20-разрядного кода необходим блок памяти объемом 2^{20} ячеек. Чтобы уменьшить этот объем, можно вместо одного применить несколько параллельно включенных блоков памяти. Как показано на рис. 7.35, *б*, 20-разрядный код из регистра RG разделен на две группы по десять разрядов. Каждая группа используется в качестве адреса для выбора одной из $2^{10} = 1024$ ячеек в соответствующих блоках памяти RAM1 и RAM2. Эти блоки осуществляют сравнение фрагментов входного и эталонных кодов.

Такое решение, однако, предполагает наличие дополнительных схем (на рисунке не показаны) для согласованной оценки результатов сравнения фрагментов кодов. Так, условие Y1 = Y2 = 0 соответствует точному совпадению входного кода с одним из эталонных только в том случае, когда 20-разрядный код X1 X2 также совпадает с этим же эталонным кодом. (Иными словами, нужно исключить из рассмотрения совпадения фрагментов входного кода с соответствующими фрагментами, принадлежащими разным эталонным кодам.) Ошибки, обнаруженные во фрагментах входного кода (коды Y1 и Y2), должны суммироваться для последующего сравнения с максимально допустимым числом.

Таким образом, в условиях повышенного уровня помех приемник проявляет определенную “терпимость” к ошибкам во флаговых кодах. Он пытается поддерживать достигнутую ранее или возобновить утраченную синхронизацию с передатчиком. Принятие решения о наличии или потере синхронизации основывается на основе анализа статистических данных об ошибках.

7.7. Поиск флага в потоке данных, передаваемых по волоконно-оптической линии связи

Задача состоит в обнаружении флаговой комбинации битов во входном потоке данных. Но специфика ее решения [19] обусловлена очень высокой скоростью передачи данных, при которой длительность битового интервала составляет доли наносекунды.

Устройство для распознавания флаговой комбинации битов (рис. 7.36) содержит оптический разветвитель (рис. 7.37), двенадцать оптических линий задержки, два фотоприемника, дифференциальный усилитель и компаратор. В данном примере флаг представлен 13-разрядным кодом 1111100110101. При обнаружении флага формируется сигнал S = 1.

Данные D передаются по волоконно-оптической линии связи в виде световых сигналов (рис. 7.38). Наличие света соответствует лог. 1, отсутствие света – лог. 0. Каждый бит размещен в отведенном ему временном интервале.

Оптический разветвитель разделяет входной световой поток на 13 равных по интенсивности частей. (На входе устройства можно установить усилитель для повышения мощности потока.) Каждая часть разделенного светового потока попадает в соответствующее выходное оптоволокно (линию). Верхняя (по схеме) линия (см. рис. 7.36) предназначена для непосредственной передачи сигнала на один из входов фотоприемника F2. Задержка передачи сигнала по этой линии условно принимается равной нулю (“нуль, умноженный на T” или 0T; T – длительность битового интервала).

Задержка передачи сигнала по второй – тринадцатой линиям составляет соответственно 1T, 2T, 3T, . . ., 12T. Задержка сигнала в каждой линии определяется числом последовательно включенных петель из оптоволокна. Каждая петля задерживает сигнал на время, равное длительности одного битового интервала T. Такая структура позволяет развернуть последовательный код в параллельный.

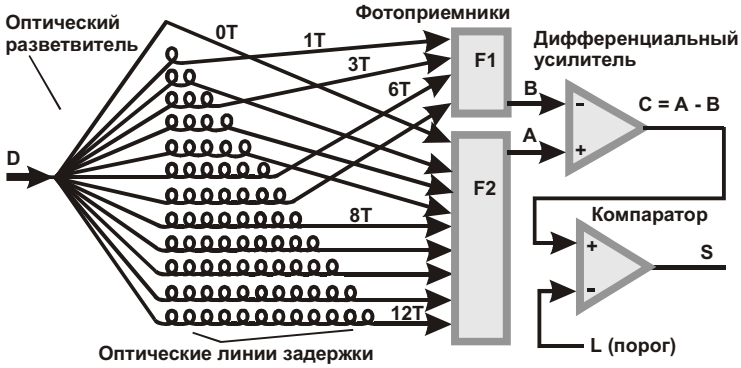

рис. 7.36. Схема устройства для распознавания флаговой комбинации битов во входном потоке данных D, поступающем по волоконно-оптической линии связи

Действительно, одновременно с поступлением новейшего бита на входы фотоприемников поступают: новый, задержанный на один такт (1T); предшествующий ему и задержанный на два такта (2T) и т. д. и, наконец, наистарейший бит, задержанный на 12 тактов (12T). Таким образом, на входах фотоприемников образуется 13-разрядное окно, через которое можно просматривать последовательность входных данных D. При этом наистарейший бит соответствует принятому на вход устройства в интервале t_0 (см. рис. 7.38), а новейший бит – принятому в интервале t_{12}.

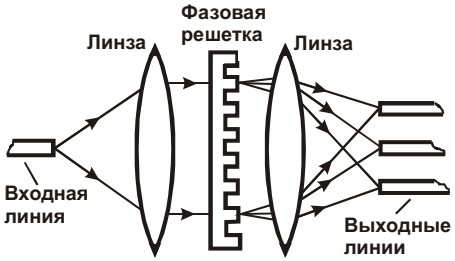

рис. 7.37. Конструкция оптического разветвителя

Выбранный в данном примере вариант подключения фотоприемников к предыдущим каскадам устройства определяется искомым флаговым кодом (1111100110101). Девять единичных битов этого кода собираются фотоприемником F2, а четыре нулевых – фотоприемником F1. Электрический сигнал на выходе фотоприемника пропорционален суммарной мощности световых потоков на его входах.

При поступлении искомого флагового кода фотоприемник F1 не получает световой энергии ни по одному из входов. Напротив, фотоприемник F2 получает световые потоки от всех входных линий. Такая комбинация сигналов уникальна и сопровождается максимально возможной разностью напряжений между точками A и B на входах дифференциального усилителя (см. рис. 7.38). Остается только зарегистрировать максимальную разность, и задача решена.

Для этого разностный сигнал C = A – B подается на компаратор и сравнивается с заданным порогом L. Сигнал опознания S = 1 формируется при превышении порога сигналом C. Порог выбирается настолько высоким, что сигнал S = 1 формируется только при точном опознании флагового кода.

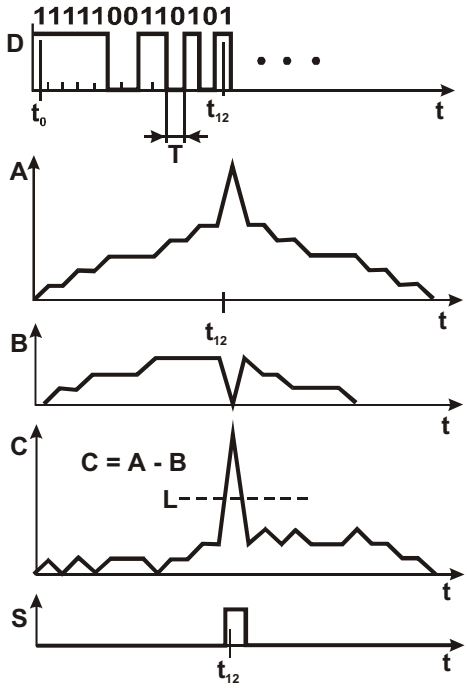

рис. 7.38. Временные диаграммы сигналов при распознавании флага

Снижение порога позволяет вести не точный, а приближенный поиск нужной последовательности битов во входном потоке. Например, сигнал S = 1 при определенном снижении порога может формироваться не только в результате поступления флагового кода (1111100110101), но и любого близкого к нему, содержащего одиночную ошибку (например, 1111100110100, ошибка в младшем разряде). Это может оказаться полезным при повышенном уровне ошибок в канале связи. Полученные результаты приближенного поиска сортируются, и, с учетом предыстории, среди них выявляются наиболее правдоподобные, периодически повторяющиеся, которые интерпретируются как флаги.

В [19] показано, что при оптимальном выборе флаговых кодов можно получить максимально контрастные картины опознания, при которых пик разностного сигнала C выражен наиболее ярко.

7.8. Передача данных вместо избыточных битов синхронизации кадра

Как уже отмечалось, данные, передаваемые по транспортной сети, группируют в структурные единицы, которые в зависимости от используемой технологии называют контейнерами, ячейками, пакетами, кадрами и т. п. В любом случае удаленный узел – приемник данных должен распознать начало передаваемой по сети структурной единицы.

Рассмотрим одну из наиболее простых структурных единиц – кадр, предусмотренный рекомендацией ITU-T V.110 (рис. 7.39, *а*).

Кадр построен в виде матрицы из десяти строк и восьми столбцов. В пересечении каждой строки и столбца размещен один бит, так что кадр содержит 80 бит. Первым передается крайний левый бит верхней строки (лог. 0), последним – крайний правый бит B63 нижней строки. Биты B1 – B63 представляют собой передаваемые данные; S1 – S8 – так называемые биты синхронизации.

В начале кадра всегда содержится флаг – кодовая комбинация из восьми лог. 0 и последующей лог. 1 (000000001). Такая комбинация не должна встречаться в любом другом месте кадра; это гарантируется введением битов S1 – S8 синхронизации, каж-

дый из которых представлен лог. 1. Из приведенного на рис. 7.39, *б* примера видно, что в каждой строке матрицы, начиная со второй, содержится по крайней мере одна лог. 1, так что флаговая комбинация просто не умещается где-либо в ненадлежащем месте. Как видим, все сделано правильно; но “работа по правилам” не всегда дает наилучший результат. Действительно, левый столбец матрицы является избыточной константой, рассчитанной на наихудший случай, когда все биты B1 – B63 представлены лог. 0 либо среди них имеется незначительное число лог. 1. Нельзя ли уменьшить избыточность левого столбца?

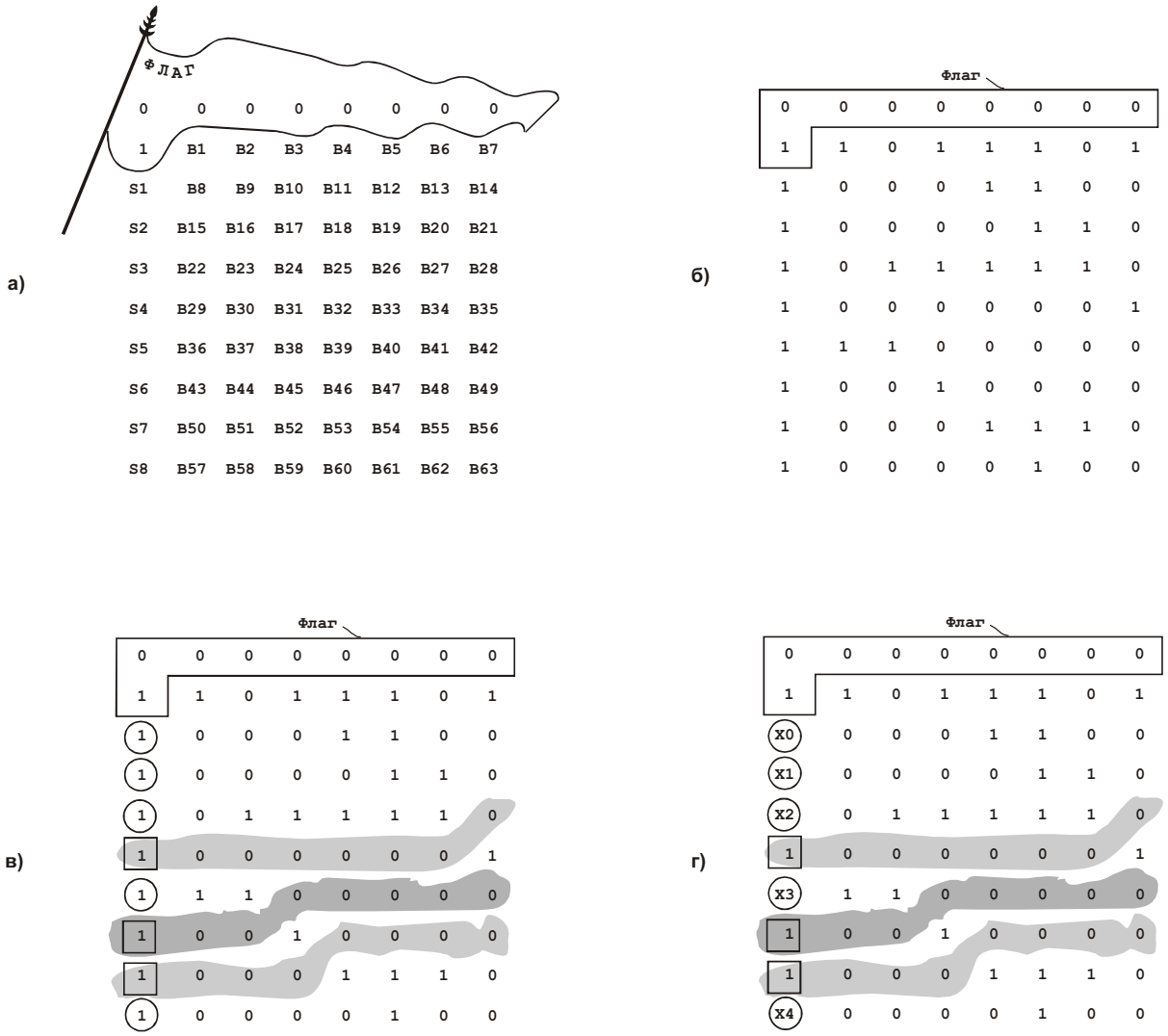

Рис. 7.39. Передача данных вместо избыточных битов синхронизации кадра: *а* – общая структура кадра в соответствии с рекомендацией ITU-T V.110; *б* – кодовый пример кадра; *в* – поиск битов синхронизации, которые должны безусловно принимать значения лог. 1 (выделены квадратиками); *г* – определение позиций для передачи данных (X0 – X4) вместо битов синхронизации (выделены кружочками)

Обратимся к рис. 7.39, *в* и проанализируем значимость битов синхронизации, начиная с первого S1 (крайнего левого бита третьей сверху строки матрицы). Он, как и предусмотрено рекомендацией V.110, равен лог. 1. Но, просматривая его ближайшее окружение, видим, что ложного опознания флага не произойдет, если биту S1 присвоить значение, равное лог. 0.

Действительно, в третьей сверху строке матрицы при S1 = 0 появится непрерывная последовательность из четырех лог. 0, но это по-прежнему гарантирует удаленный

приемник кадра от ложного опознания флага (не только на своем законном месте). Отметим факт свободы выбора значения бита S1, выделив его на рисунке кружочком.

Подобные рассуждения применимы также к битам S2, S3, S5 и S8, выделенным на рисунке кружочками. Эти биты могли бы принимать не только единичные, но и нулевые значения, не мешая удаленному приемнику распознать флаг только на своем законном месте – в начале кадра. В то же время биты S4, S6 и S7 (выделены квадратиками) должны безусловно принимать значения лог. 1. В противном случае сформируются достаточно длинные последовательности из лог. 0 (выделены тремя затененными областями), которые, например, при S4 = S6 = S7 = 0 приведут к троекратному ложному опознанию начала кадра в его теле.

Таким образом, становится понятной идея, предложенная в [3]. Ее можно сформулировать так: в передаваемом кадре V.110 обычно можно отыскать вакантные места для размещения от одного до восьми битов – своего рода “неучтенных пассажиров” транспортной системы. В нашем примере имеем пять таких битов X0 – X4 (рис. 7.39, *г*).

Теперь возникает вопрос: как удаленный приемник сможет понять, какие биты полученного кадра обведены кружочками, а какие – квадратиками? Ведь информация о форме обвода в явном виде не передается по линии! Попробуем с этим разобраться.

Алгоритм работы приемника таков.

1. Приемник, постоянно прослушивая линию, обнаруживает поступление флага (кода 000000001), после чего принимает кадр (см. рис. 7.39, *а*) и записывает его в первой области памяти, выделенной для временного хранения кадров.

2. Приемник запоминает биты S1 – S8 во второй области памяти, выделенной для их временного хранения, а на места этих битов в матрице помещает лог. 1. Таким образом, приемник “видит” матрицу, показанную на рис. 7.39, *б*.

3. Приемник проводит описанный ранее анализ необходимости существования каждого бита синхронизации и, если не было ошибок передачи данных B1 – B63, приходит к тем же результатам, которые были приведены на рис. 7.39, *в*. В нашем примере приемник приходит к выводу, что биты S4, S6 и S7 должны быть представлены лог. 1.

Он проверяет, так ли это, извлекая из второй области памяти соответствующие биты. (Любое несовпадение расценивается как ошибка передачи, и кадр отбрасывается.) Попутно выявляются позиции битов, обведенных кружочками. Теперь остается только извлечь из второй области памяти соответствующие биты (S1 – S3, S5, S8) и скомпоновать из них искомый код X0 … X4 (см. рис. 7.39, *г*). Таким образом, “неучтенные пассажиры” благополучно прибыли на место.

Можно предложить много вариантов использования кодов X0 …Xi. Например:

1) коды X0 … Xi переменной длины, передаваемые в последовательности кадров, можно рассматривать как фрагменты некоторого, в общем случае независимого, массива данных. Иными словами, можно организовать второй, параллельный канал передачи данных, который работает не в ущерб первому, основному;

2) каждый код X0 … Xi можно рассматривать как некое контролирующее приложение к данным B1 – B63, которые передаются в этом же кадре. Например, разряд X0 может представлять собой бит контроля по четности всей группы B1 – B63. Разряды X1 и X2 могут соответственно служить битами контроля по четности подгрупп B1 – B31 и B32 – B63 и т. д. с уменьшением длины контролируемой области.

Иное возможное распределение функций разрядов: бит Xi контролирует по четности данные, размещенные в своей и предыдущей строках матрицы. Такое решение позволяет обнаруживать ошибки “на лету”, не дожидаясь окончания приема полного кадра.

Возможны варианты использования кодов X0 … Xi для уменьшения вероятности ложного опознания флага при наличии ошибок передачи [3].

7.9. Способы размещения низкоскоростного потока данных в высокоскоростном потоке кадров

Для передачи данных от некоторого источника, например, от низкоскоростного аналого-цифрового преобразователя (АЦП), к удаленному регистрирующему устройству, например, компьютеру, может использоваться высокоскоростной канал связи (рис. 7.40). Канал сдержит передающую и приемную аппаратуру (например, два мультиплексора), подключенную к противоположным сторонам линии связи. Разумеется, канал связи передает потоки данных и от других источников (на рисунке не показаны). Задача состоит в рациональном размещении низкоскоростного потока данных D в высокоскоростном потоке кадров, передаваемых по линии. Рассмотрим два способа решения этой задачи [29].

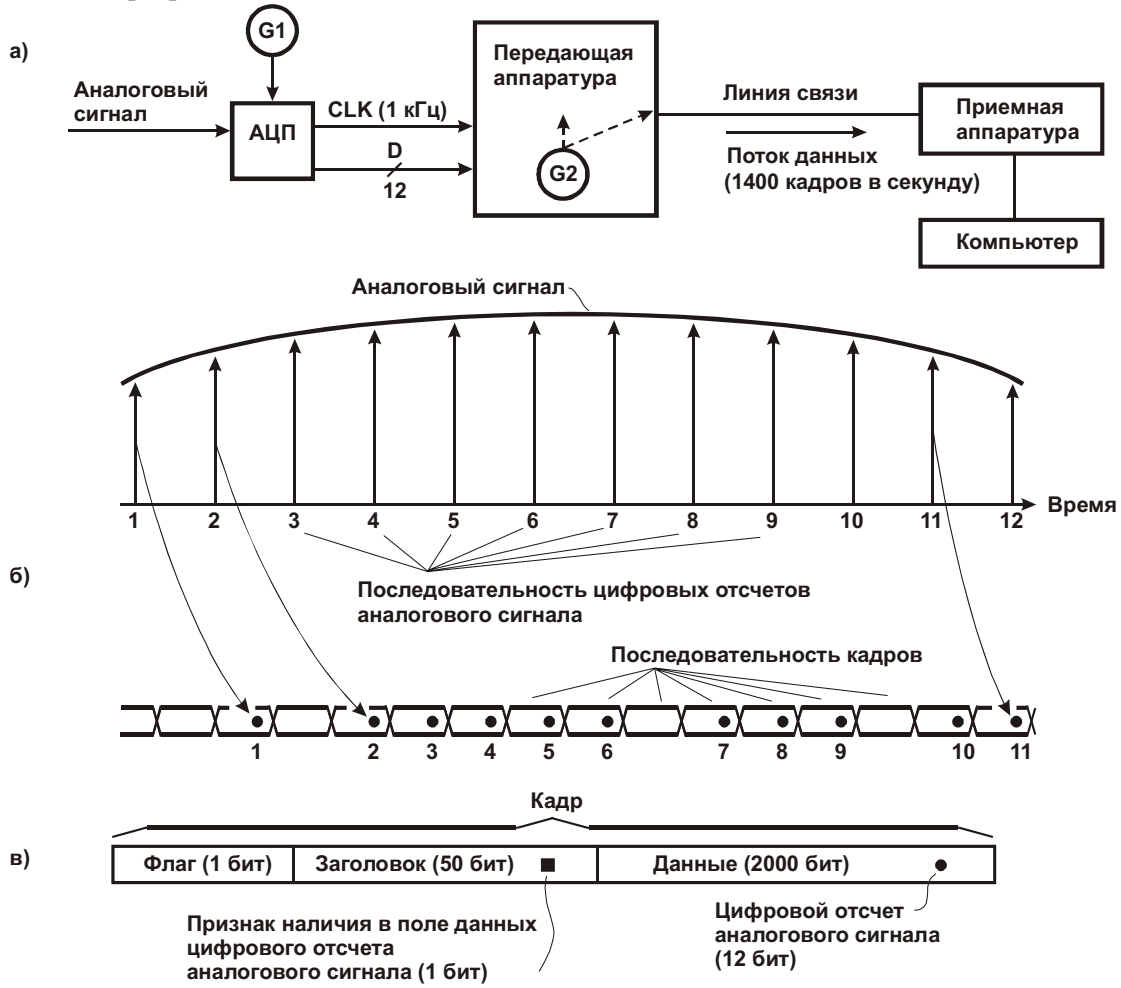

Рис. 7.40. Передача низкоскоростного потока данных от аналого-цифрового преобразователя к удаленному компьютеру (вариант 1): *а* – схема; *б* – размещение потока данных в потоке кадров; *в* – структура кадра

Первый способ. Как показано на рис. 7.40, АЦП формирует непрерывную последовательность 12-разрядных цифровых отсчетов D аналогового сигнала. Синхронизация АЦП осуществляется от генератора синхросигналов G1. Отсчеты следуют с частотой 1кГц и сопровождаются синхросигналом CLK.

Передающая аппаратура синхронизируется сигналом от генератора G2. Данные поступают в линию связи в виде непрерывной последовательности кадров со скоро-

стью 1400 кадров в секунду. Кадр содержит одноразрядный флаг (см. п. 7.2.1), обозначающий его начало, 50-разрядный заголовок со служебными данными и поле данных (2000 бит).

Передающая аппаратура принимает очередной отсчет D аналогового сигнала и размещает его в определенной группе разрядов поля данных ближайшего формируемого кадра. В каждый кадр может быть помещен только один отсчет. Факт его размещения помечается установкой в состояние лог. 1 определенного бита в заголовке кадра. Как следует из рис. 7.40, *б*, отсчеты переносятся в поток кадров с нерегулярным темпом, по мере их поступления от АЦП и наличия "свободных мест" в транспортной системе.

Приемная аппаратура анализирует поступающие кадры. При обнаружении в заголовке кадра признака наличия цифрового отсчета аналогового сигнала соответствующий 12-разрядный код извлекается из поля данных и передается в компьютер для дальнейшей обработки. Если кадр не содержит данных от АЦП, то отведенное для них место используется для передачи данных от других источников.

Второй способ. Передача данных от АЦП в компьютер может синхронизироваться одним генератором G (рис. 7.41), при этом скорость поступления данных, как и в предыдущем примере (рис. 7.40, *а*), не совпадает со скоростью потока кадров. Однако в данном случае возможно регулярное размещение данных в потоке кадров. Это означает, что в каждой группе из M кадров размещаются N цифровых отсчетов аналогового сигнала. В примере, приведенном на рис. 7.41, поток разбивается на группы по девять кадров (M = 9), а в каждой группе размещаются четыре отсчета аналогового сигнала (N = 4).

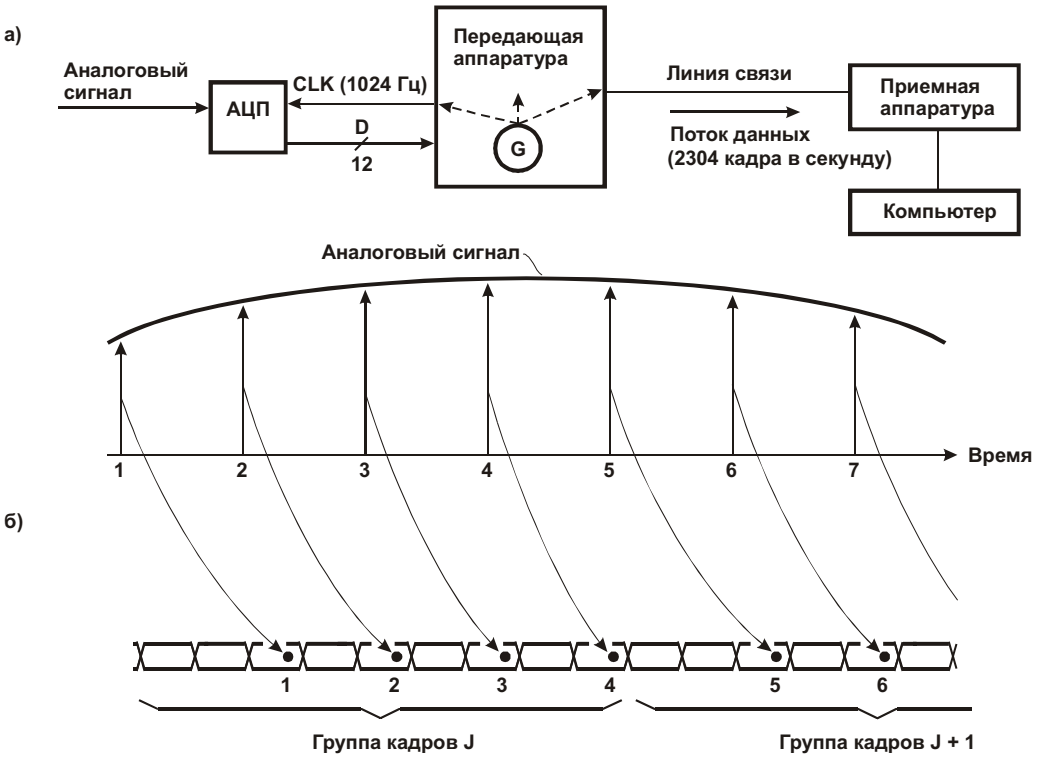

Рис. 7.41. Передача низкоскоростного потока данных от аналого-цифрового преобразователя к удаленному компьютеру (вариант 2): *а* – схема; *б* – размещение потока данных в потоке кадров

Убедимся в правильности такого разбиения. За одну секунду по линии связи передаются 2304 кадра или 256 групп по девять кадров. В каждой группе содержатся четы-

ре отсчета, поэтому их общее число составит $256 \times 4 = 1024$, что соответствует скорости поступления данных от АЦП.

В общем случае при некратном отношении частоты F_F следования кадров к частоте F_S следования цифровых отсчетов аналогового сигнала выполняется условие $F_F/F_S = A + b/c$, где $F_F/F_S > 1$, A – целая часть отношения частот, b/c – несократимая дробь. Это условие можно записать в следующем виде: $F_F/F_S = (Ac + b)/c$, где $Ac + b = M$, $c = N$.

Разметку потока кадров на группы можно выполнить по меньшей мере двумя способами.

1. Первый кадр каждой группы может помечаться установкой в единичное состояние некоторого служебного бита в его заголовке. Приемная аппаратура, обнаружив признак начала группы кадров, получает ориентир для поиска и выделения данных от АЦП. Это возможно благодаря тому, что размещение этих данных в каждой группе кадров одинаково и заранее известно. Так, в приведенном на рис. 7.41 примере первый и второй кадры каждой группы не содержат данных от АЦП, в третьем кадре данные имеются и т. д.

2. Для более экономной разметки потока кадров на группы можно использовать псевдослучайные одноразрядные флаги (см. п. 7.2.2). Экономия заключается в том, что применение псевдослучайной последовательности флагов позволяет приемной аппаратуре находить начало группы кадров только на основе анализа этой последовательности. При этом нет необходимости помечать первый кадр группы служебным битом в заголовке.

Применительно к рассматриваемой системе передачи данных (рис. 7.41) следует генерировать псевдослучайную последовательность битов с периодом повторения, равным или кратным девяти. Точнее, последовательность не обязательно должна быть псевдослучайной в строгом смысле слова (см. п. 8.4.1). В данном случае достаточно принять за основу классическую псевдослучайную последовательность с периодом повторения, равным $2^4 - 1 = 15$ бит и выделить из нее только девять следующих подряд элементов, отбросив шесть остальных. Один из вариантов формирования такой “усеченной” последовательности поясняется рис. 7.42.

После начальной установки в регистре RG некоторого ненулевого кода (цепи установки на рисунке не показаны) классический генератор псевдослучайной битовой последовательности (рис. 7.42, *а*) может находиться в 15 состояниях (см. таблицу в левой верхней части рисунка), которые периодически повторяются. Копии этих состояний можно получить при просмотре выходной последовательности битов через четырехразрядное движущееся окно, как показано в правой верхней части рисунка.

Чтобы исключить “лишние” состояния регистра RG, в схему генератора введены дополнительные элементы (рис. 7.42, *б*). Элемент И формирует сигнал лог. 1 при обнаружении в регистре кода $0011_2 = 3_{10}$. В начале следующего такта, по положительному фронту сигнала CLK триггер D1 устанавливается в единицу, что вызывает безусловный переход регистра в состояние 1111 = 15. При переходе синхросигнала CLK в состояние лог. 0 устанавливается в единицу триггер D2, что вызывает установку в нуль триггера D1 и снятие сигнала установки кода 1111 с входа регистра. В начале следующего такта в регистре формируется код 1110 = 14, а в середине этого такта сигнал лог. 0 переписывается из триггера D1 в триггер D2.

В последующих тактах регистр последовательно проходит состояния 12, 8, 1, 2, 4, 9, 3, затем описанный цикл генерации кодов повторяется. В результате число различимых состояний регистра сокращено до девяти, что и требовалось. Отметим, что в данном случае последовательность состояний регистра не соответствует кодам, наблюдаемым при просмотре выходной последовательности битов через движущееся четырехразрядное окно. Однако это не приводит к каким-либо неопределенностям, так как эти коды можно однозначно сопоставить с состояниями регистра.

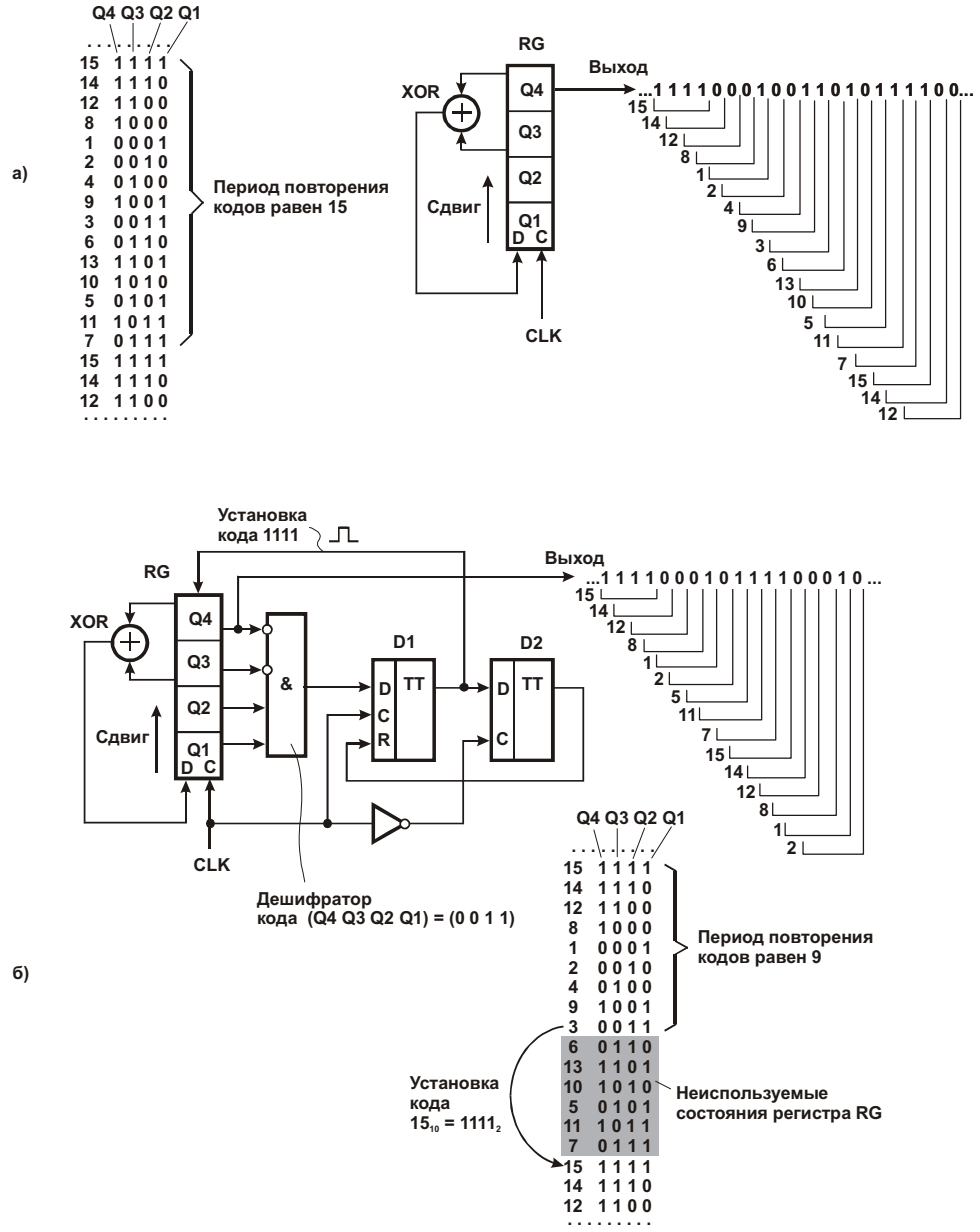

Рис. 7.42. Генераторы, формирующие: *а* – псевдослучайную битовую последовательность максимальной длины с периодом повторения, равным 15; *б* – битовую последовательность с периодом повторения, равным 9

Последовательность генерируемых таким способом битов размещается передающей аппаратурой во флаговых позициях формируемых кадров – по одному биту на каждый кадр. Эта последовательность выделяется из кадров приемной аппаратурой и просматривается ею через четырехразрядное окно (см. рис. 7.42, *б*). Согласно протоколу обмена, один из кодов в окне, например код 1111, является признаком начала группы из девяти кадров. Обнаружив такой код, приемная аппаратура “узнаёт”, что последний принятый кадр является первым в группе (возможны и иные договоренности). Разумеется, что передающая аппаратура должна знать, что к этому моменту она действительно переслала именно первый кадр группы. Таким образом, осуществляется кадровая синхронизация между приемной и передающей аппаратурой.

Рассмотренные решения позволяют согласовать темп поступления данных от низкоскоростного источника с темпом их передачи по высокоскоростной транспортной системе. Для такого согласования можно выделить один служебный бит в каждом кад-

ре. Но можно и не вводить служебные биты, если использовать псевдослучайные флаги начала кадров.

7.10. Уменьшение числа операций, выполняемых при распознавании флага начала кадра

Распознавание многоразрядных флагов начала кадра в потоке данных обычно реализуется с помощью микропроцессорного устройства. Это устройство выполняет некоторую циклическую программу, предусматривающую ряд операций извлечения данных из буферной памяти и их сопоставления с заранее известным флаговым кодом. Уменьшение общего числа таких операций позволяет снизить тактовую частоту микропроцессора и уменьшить интенсивность считывания данных из буферной памяти, что, в свою очередь, снижает потребляемую устройством мощность.

Рассмотрим один из способов уменьшения числа операций, выполняемых при распознавании флага начала кадра [41]. Чтобы показать преимущество этого способа перед традиционным, предположим, что тот и другой реализуются с использованием одной и той же аппаратуры обработки сигнала, поступающего из линии (рис. 7.43, *а*). Прежде всего, из этого сигнала выделяется битовый поток данных DATA и сопровождающий его синхросигнал CLK. Далее поток битов преобразуется в поток байтов D, сопровождаемый синхросигналом CLK/8 с выхода делителя частоты на восемь. Поток байтов поступает в буферную память и считывается из нее для дальнейшей обработки (см. штриховую линию на рисунке). Устройство распознавания флага выполнено на основе микропроцессора. Оно анализирует проходящий через буферную память поток данных и при обнаружении флага начала кадра формирует сигнал FOUND.

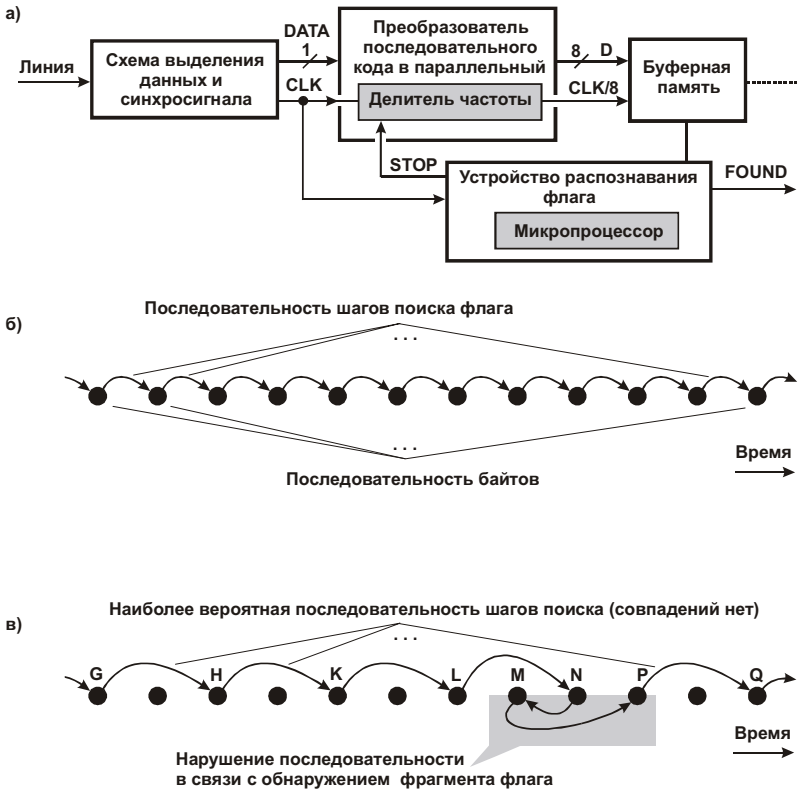

Рис. 7.43. Распознавание флага начала кадра: *а* – структурная схема аппаратуры обработки сигнала из линии; *б, в* – диаграммы, соответствующие традиционному и предлагаемому способам поиска

Исходное состояние делителя частоты может быть произвольным, поэтому первоначальное разделение битового потока данных DATA на байты D, вероятнее всего, неверно, т. е. не совпадает с разделением, выполненным удаленным передатчиком данных в линию. Следствием этого является невозможность регулярного обнаружения флагов начала кадров в потоке байтов D, что достижимо только при правильном определении границ байтов.

Чтобы найти истинные границы байтов, устройство распознавания флага проводит серию экспериментов по поиску регулярной последовательности флагов. В каждом эксперименте границы байтов смещаются на один бит, поэтому число экспериментов не превышает семи, а при удачном стечении обстоятельств сразу же выясняется, что предполагаемые границы байтов совпадают с истинными границами. Для того чтобы сместить границы байтов на один бит, устройство распознавания флага формирует сигнал STOP, который приостанавливает работу делителя частоты на один такт сигнала CLK.

Предположим, что границы байтов в битовом потоке определены правильно, а флаг представлен 32-разрядным кодом

$$0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1011\ 0011_2 = 00\ 00\ 01\ B3_{16}.$$

Традиционный способ поиска флага предписывает микропроцессору последовательно просматривать байты, размещенные в буферной памяти (рис. 7.43, *б*) и сравнивать их с байтами кода $00\ 00\ 01\ B3_{16}$, начиная с первого (00_{16}). Успешное распознавание флага сопровождается последовательным совпадением четырех байтов данных с байтами этого кода.

Предлагаемый способ поиска предусматривает проведение беглого оценочного просмотра последовательности байтов. Просмотр становится более детальным только в окрестностях тех точек (байтов), которые подозреваются на причастность к переносу флага (рис. 7.43, *в*). В результате число выполняемых операций, условно показанных на рисунке стрелками, уменьшается, что и требовалось. Рассмотрим оба способа поиска подробнее.

Традиционный способ поиска флага описывается диаграммой состояний, приведенной на рис. 7.44. Стрелками помечены разрешенные переходы между состояниями. Блок-схема алгоритма поиска показана на рис. 7.45. Предположим, что в начале поиска устройство находится в состоянии 0. Если очередной байт равен 00_{16}, то осуществляется переход в состояние 1, в противном случае сохраняется состояние 0. При повторном обнаружении байта 00_{16} происходит переход в состояние 2, в противном случае – в состояние 0 и т. д. Эти же условия соответствуют переходам между блоками 1 – 4 на блок-схеме алгоритма. Обнаружение флага соответствует переходу в состояние 4 или, что то же самое, выходу из блока 8 по ветви “Да”.

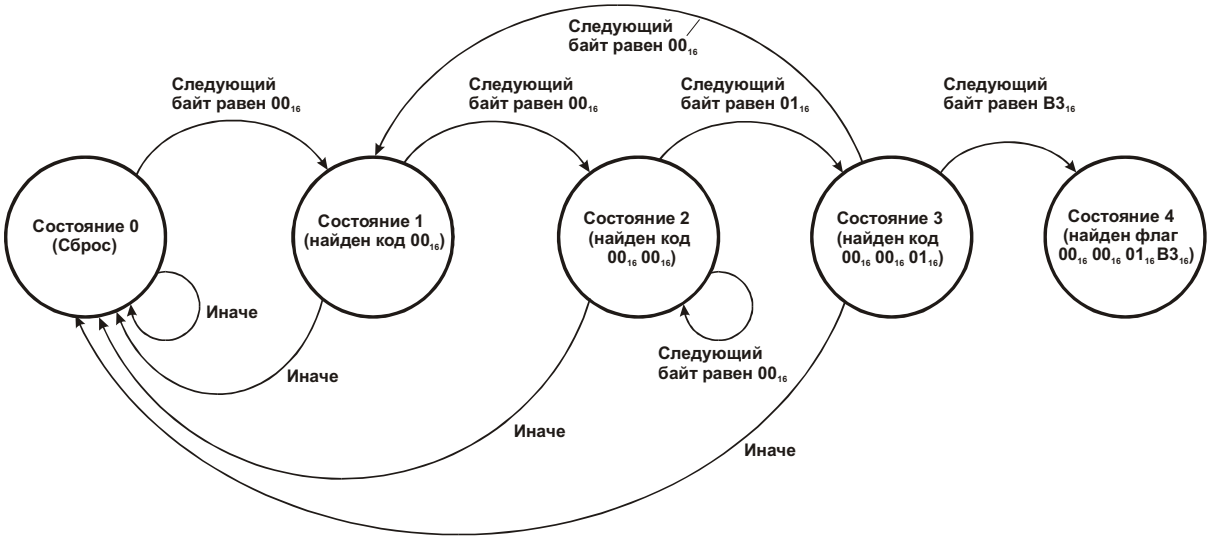

Рис. 7.44. Диаграмма состояний устройства распознавания флага (традиционный способ поиска)

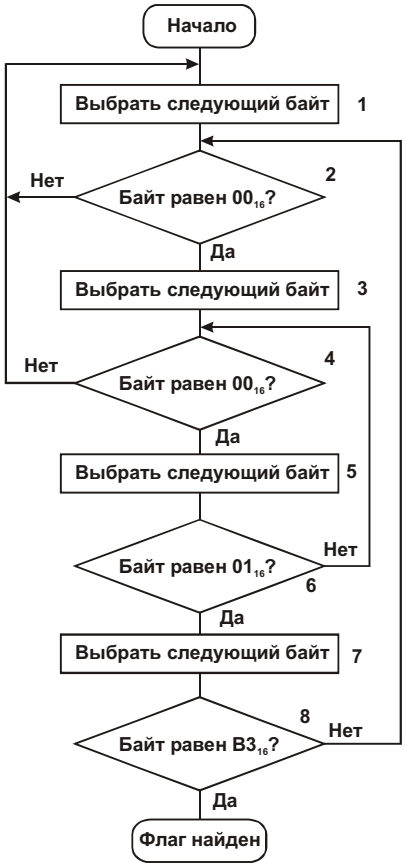

Рис. 7.45. Блок-схема алгоритма поиска флага (традиционный вариант)

Диаграмма состояний и блок-схема предлагаемого алгоритма поиска флага приведены на рис. 7.46 и рис. 7.47.

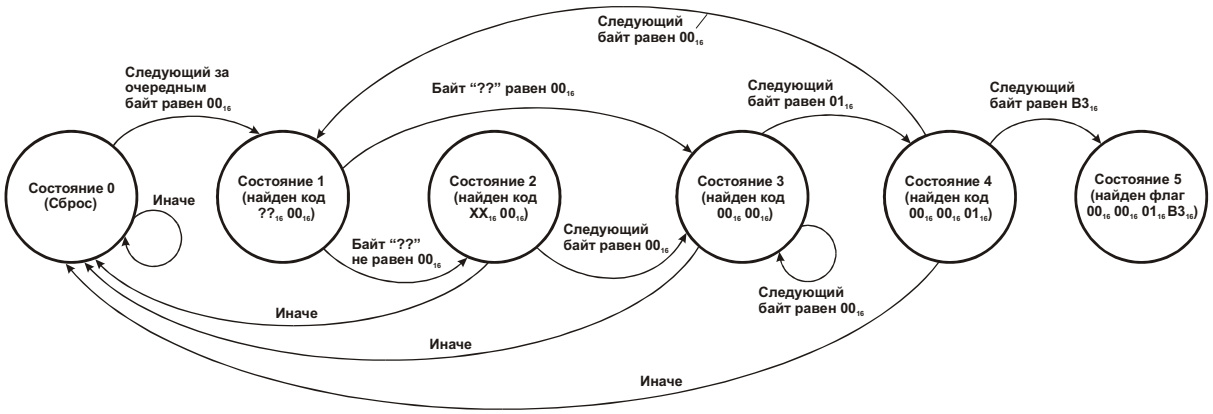

Рис. 7.46. Диаграмма состояний устройства распознавания флага (предлагаемый способ поиска)

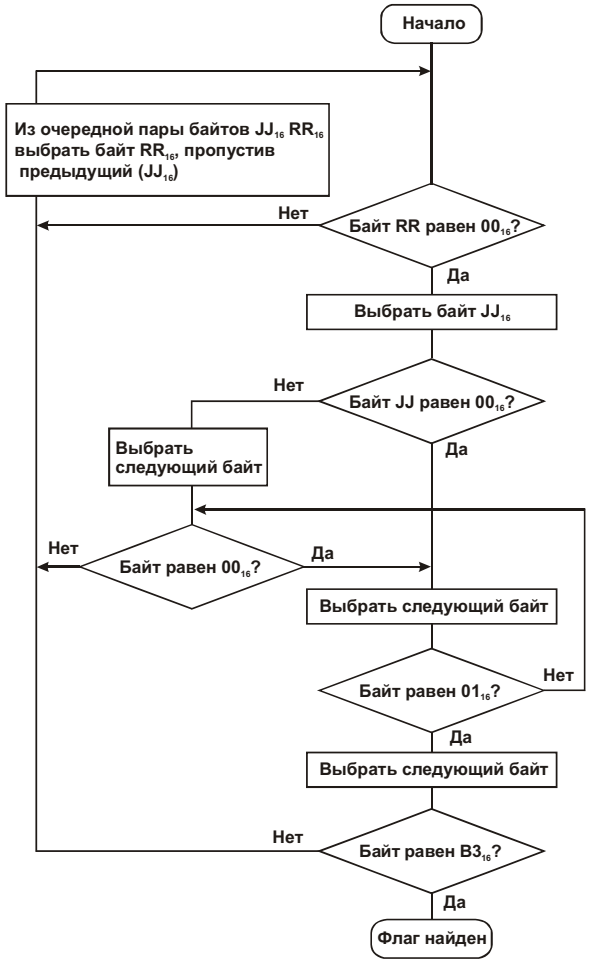

Рис. 7.47. Блок-схема алгоритма поиска флага (предлагаемый вариант)

Отличие предлагаемого способа поиска от традиционного, как уже отмечалось, состоит в том, что вместо полного контроля последовательности байтов проводится ее грубый просмотр, а подробности анализируются лишь по мере необходимости уточнения ситуации. Такой способ поиска правомерен в силу следующих причин.

1. Флаги встречаются в потоке данных сравнительно редко, особенно если они представлены уникальными кодами (в случае применения битстаффинга), а кадры содержат большое число байтов. Поэтому столь же редко возникает необходимость полного анализа четырехбайтового кода в связи с обнаружением флага.

2. Необходимым (но не достаточным) условием обнаружения флага является регистрация двух размещенных рядом нулевых байтов. Поэтому для проверки этого условия можно проверять не все байты подряд, а продвигаться по их последовательности с шагом, равным двум байтам. Данные в кадрах можно рассматривать как случайные – это безусловно справедливо, если применено их скремблирование. Тогда вероятность обнаружения нулевого байта равна 1/256. Иными словами, ускоренное продвижение по последовательности байтов – скорее норма, чем исключение.

Для пояснения сказанного вернемся к диаграмме, показанной на рис. 7.43, *в*. В процессе поиска флага последовательно анализируются ненулевые байты G, H, K и L, расположенные в буферной памяти с интервалом в две ячейки. На диаграмме состояний (рис. 7.46) этот этап поиска соответствует неудачным попыткам выхода устройства из состояния 0. После байта L выбирается нулевой байт N, что соответствует переходу устройства в состояние 1. При этом возникает необходимость проверить предыдущий байт M, так как появилась некоторая надежда обнаружить флаг. Однако этот байт при проверке оказывается ненулевым, устройство переходит в состояние 2. Далее выбирается байт P, но он также оказывается ненулевым, поэтому устройство возвращается в состояние 0. После этого ускоренный анализ продолжается – выбирается байт Q и т. д.

Таким образом, предлагаемый способ значительно сокращает число обращений к памяти и уменьшает объем вычислительных операций, выполняемых микропроцессором при поиске флага.