УДК 681.327+656.34

РАЗРАБОТКА СИСТЕМ ПЕРЕДАЧИ ТЕЛЕМЕТРИЧЕСКОЙ ИНФОРМАЦИИ ДЛЯ КОСМИЧЕСКИХ ОБЪЕКТОВ

В.П. ЗИНЧЕНКО, В.А. БУРОВ, С.В. ЗИНЧЕНКО, А.В. ШТЕФЛЮК

Рассмотрены вопросы разработки космических объектов в соответствии с протоколами Консультативного комитета по космическим информационным системам (CCSDS). Описаны методы кодирования информации телеметрического кадра, структуры кодеров и декодеров, рандомизатора формирователя телеметрического кадра, обладающих полнотой, достаточной для моделирования формирователя телеметрического кадра на основе VHDL-проекта с последующей загрузкой в программируемые логические схемы XCV 300E-PQ240 и отладкой.

ВВЕДЕНИЕ

При проектировании космических объектов применяются протоколы, позволяющие реализовать международные соглашения по взаимодействию открытых систем [1]. Это вызвано тем, что одни космические объекты предназначены для сбора информации и ее передачи, другие — еще и для приема информации (управляющие команды). Поэтому для объектов первого типа создаются проекты систем передачи телеметрической (ТМ) информации, для второго — передачи и приема информации. С учетом тенденции микроминиатюризации космических объектов (микро- и наноспутники) [2] такие системы целесообразно проектировать на одном программируемом кристалле (SOPC) вместо многоплатных конструкций или вычислительных комплексов на микроконтроллерах [3].

Для космической промышленности характерно мелкосерийное производство специализированных интегральных схем (СпИС), для которых не характерно полностью заказное проектирование ввиду большой его стоимости. Этим объясняется применение интегральных схем программируемой структуры (ИСПС). Отметим, что ИСПС также позволяет корректировать макеты и опытные партии и обеспечивает секретность разработок, так как структура ИСПС невидима в отличие от структур на стандартных дискретных элементах.

Разделение космических объектов по признаку передачи/приема информации доказывает целесообразность разработки канала передачи телеметрической информации на одной ИСПС, а канала приема — на другой.

Желательно также снабдить эти ИСПС интерфейсами для тестирования JTAG (Joint Test Automation Group).

Постановка задачи сводится к проектированию систем, реализующих передающий канал, в которых предусматриваются синхронизация телеметрической информации и кодирование с помощью кодов, исправляющих ошибки. Описанные разработки согласованы с рекомендациями Консультативного комитета по космическим информационным системам (CCSDS) [4].

Рекомендации CCSDS по созданию космических информационных систем. В таблице показаны соотношения между протоколами и слоями в открытых системах (OSI — Open System Interconnection) [1].

Отношения между слоями

Слои OSI	**Слои CCSDS**	**Протоколы CCSDS**
Сеть и верхние слои	Сеть и верхние слои	
Связывающий слой данных	Подслой протокола связи данных	Космический TM-протокол связи данных
	Подслой синхронизации и кодирования канала	Синхронизация и кодирование TM-канала
Физический слой	Физический слой	Радиочастотные и модуляционные системы

Подслой синхронизации и кодирования канала выполняет три функции: кодирование с исправлением ошибок, включая проверку фрейма; синхронизацию; псевдорандомизацию (опционно). В CCSDS также рекомендовано три типа кодов, исправляющих ошибки: сверточные коды; коды Рида–Соломона; турбокоды [5].

Для синхронизации фрейма передачи рекомендуется присоединять к началу кодового блока синхромаркер (ASM−Attached Sync Marker), который также может устранять неоднозначности в данных, если они не устранены на физическом уровне. В нем может быть применен метод псевдорандомизации, который используется для повышения плотности передачи символа.

В рекомендациях предлагается структура канала передачи, содержащая его внутреннюю организацию синхронизации и кодирования. Определяются функции подслоев и логическая взаимосвязь этих функций. Протоколом подслоя связи данных канала передачи (рис. 1) принимается передача фреймов фиксированной длины, определяются заданные функции и посылается последовательный и непрерывный поток символов в физический слой.

Проектирование устройств канала передачи. Телеметрический кадр (ТК) в соответствии с рекомендациями CCSDS должен состоять из начального ASM и следующего непосредственно за ним блока данных. Шаблон синхромаркера для данных без кодирования состоит из 32 бит: 1ACFFC10h, 0001 1010 1100 1111 1111 1100 0001 1101b.

Размер блока данных без кодирования всегда кратен любому целому числу байт, но не должен превышать 2048 байт. Размер ТК в течение фазы передачи/приема не изменяется и определяется выбором на предыдущей фазе.

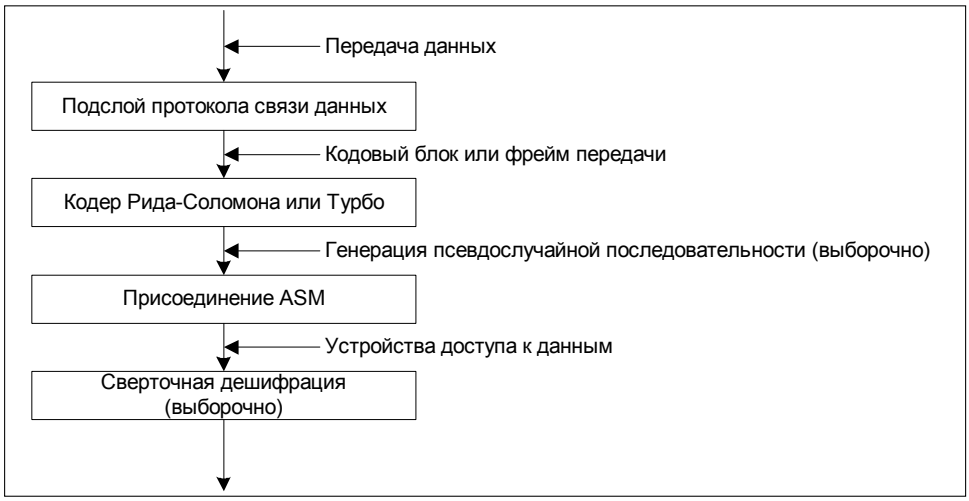

Рис. 1. Внутренняя организация подслоя передачи информации

Структурная схема каналов передачи/приема, составленная в соответствии со спецификацией CCSDS 131.0-B-1, показана на рис. 2. На вход подается бинарный код, который кодируется одним из трех кодеров. Далее сигнал подается на мультиплексор (MUX), затем на сигнал накладывается «шум». Блок выдачи синхроимпульсов (БВС) формирует управляющие сигналы для формирователя блока данных (ФБД), после чего информация поступает непосредственно в канал передачи.

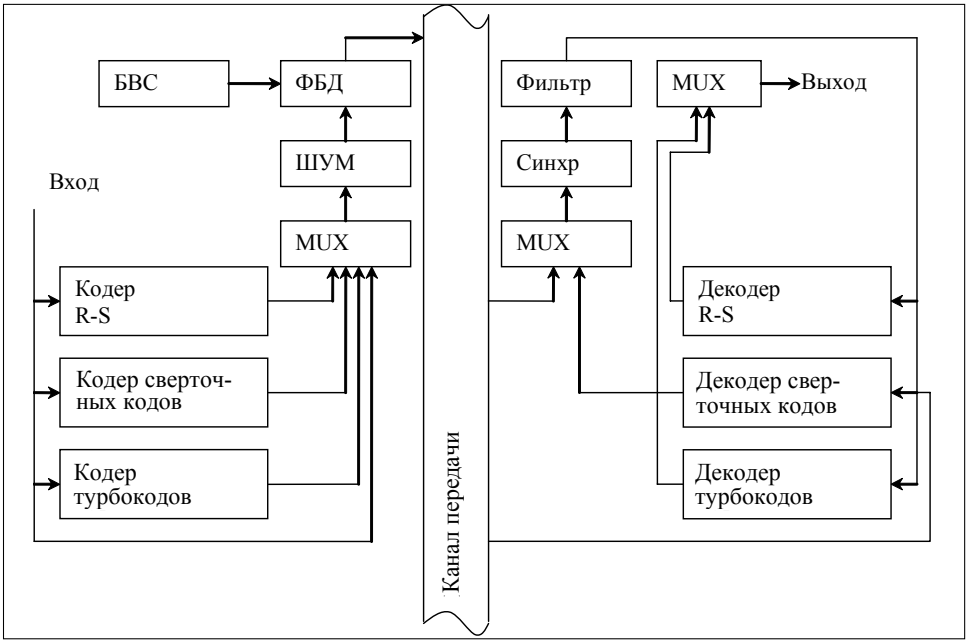

Рис. 2. Структурная схема канала передачи информации

При приеме сигнал поступает на MUX или декодер (Convolutional), затем на синхронизирующее устройство (Синхр), далее на фильтр, после чего на один из двух декодеров в соответствии с кодирующим алгоритмом и через MUX на выход.

VHDL-модель формирователя ТК. Формирователь ТК моделировался в среде Active-HDL. В результате получены временные диаграммы работы ТК (рис. 3).

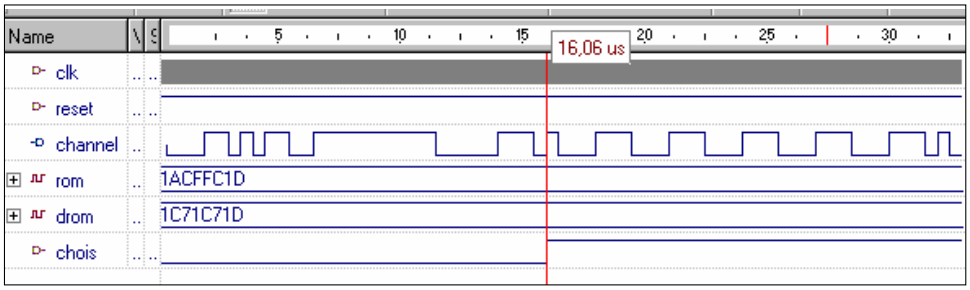

Рис. 3. Диаграммы сигналов формирователя ТК

Диаграмма сигнала синхромаркера наблюдается слева от вертикальной черты в строке channel, а в этой же строке справа — диаграмма данных. Соответствующие 16-ричные коды можно видеть в строках rom и drom. Частота синхроимпульсов clk — 2 МГц.

Кодер для сверточного кодирования. Рассматривается базовый алгоритм сверточного кодирования, который хорошо подходит для каналов с преобладанием гауссовых шумов. Ошибки при кодировании по данному методу несущественны и фактически не влияют на работу канала передачи. Процесс декодирования по алгоритму сверточного кодирования соответствует алгоритму максимального правдоподобия (алгоритм Витерби). Отметим, что этот алгоритм, применяемый отдельно, не может гарантировать достаточную вероятность передачи символа при использовании мультиплексорных схем. Поэтому в тех случаях, когда проектировщик системы не проверяет, достаточна ли плотность передачи символов другими средствами, требуется применение псевдорандомизатора.

Базовый алгоритм сверточного кодирования является несистематическим алгоритмом, определяющим процедуру декодирования с такими характеристиками:

- использует кодирование с максимальной вероятностью расшифровки;
- кодовая скорость (r) — 1/2 бита на символ;
- длина ограничения (K) — 7 бит;
- векторы связи — $G1 = 1111001b\,(39h)$; $G2 = 1011011\,(5bh)$;
- инверсия символа — на дорожке выхода $G2$;
- последовательность символов на выходе — $C_1(1)$, $\overline{C_2}(1)$, $C_1(2)$, $\overline{C_2}(2)$.

При системах модуляции, подавляющих несущую, возможно использование алгоритмов «Без возврата на нулевую отметку» (NRZ-M) или «Без возврата на нулевой уровень» (NRZ-L) как форм волны модуляции. Преобразование форм волны модуляции из NRZ-L в NRZ-M, выполняется подачей бита со входа на блок, реализующего сверточный алгоритм кодирования. Соответственно, преобразование из NRZ-M в NRZ-L выполняется перед выходом блока, реализующего сверточный алгоритм декодирования, что позволяет избежать потерь пропускной способности канала передачи.

Рассмотрим схему алгоритма (рис. 4.) Блок *D* является задержкой для каждого бита на входе, для чего генерируются два бита циклическим переключением ключа *S*1: положение 1, положение 2. Ключ *S*1 в положении 1 ассоциирован с первым битом на входе.

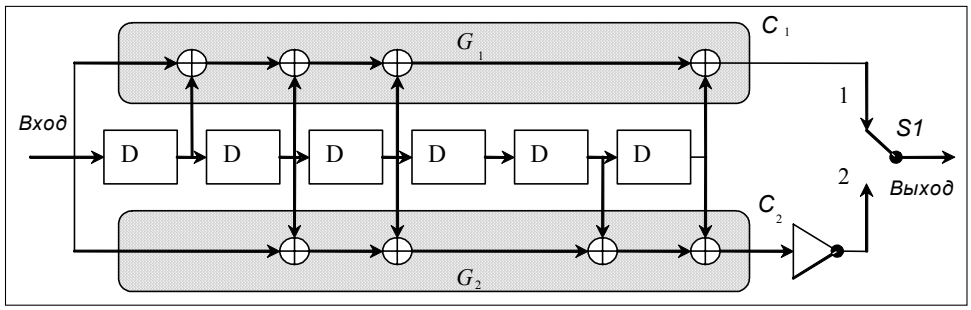

Рис. 4. Схема сверточного базового алгоритма

Проект сверточного кодера состоит из двух файлов: *Bit_Delay.vhd* и *Main_Coding.vhd*. Первый выполняет задержку каждого бита в каждом из шести блоков задержки. Особенностью является то, что на каждый блок и ключ задержки приходит синхроимпульс с частотой 100 МГц.

Сигналы *SingleBitIn* и *SingleBitOut* — входные и выходные сигналы ячеек задержки, а *Sync* играет роль синхроимпульса. Кроме того, присутствует дополнительная переменная *InsideVariable* для моделирования задержки. Результат выполнения программы *Bit_Delay.vhd* и посылка кода 11001011010 на вход одной ячейки будут такими, как показано на рис. 5.

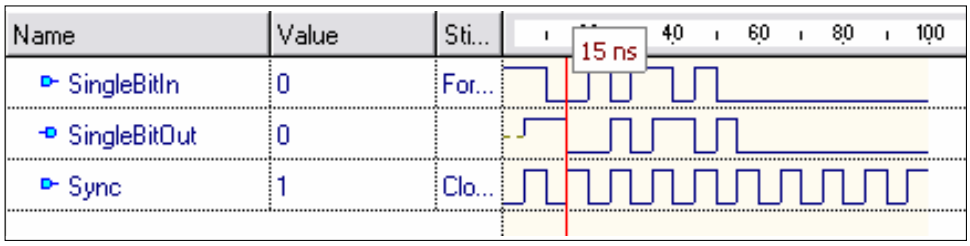

Рис. 5. Выполнение задержки

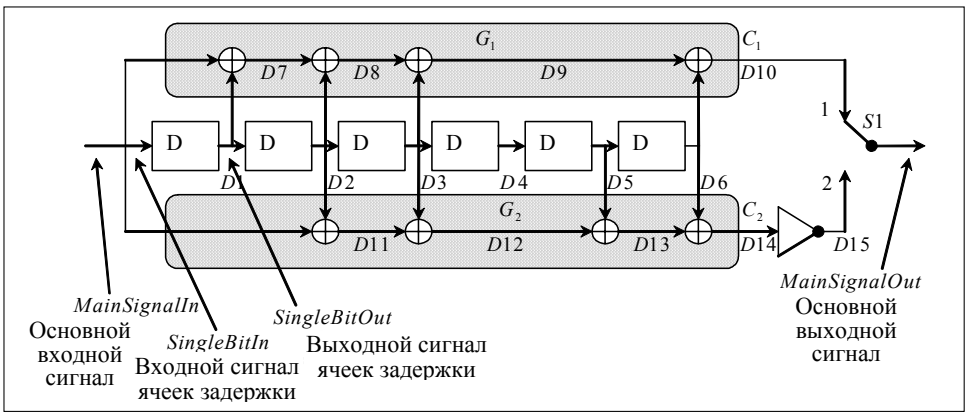

Рис.6. Имена сигналов

Как видно из рис. 5, бит информации передается с задержкой 5нс при частоте синхроимпульсов 100 МГц.

MainSignalIn и *MainSignalOut* — входные и выходные сигналы. MainSync играет роль синхроимпульса. Сигналы Д1÷Д15 распределены в соответствии с рис. 6. Результат выполнения программы при посылке кода 11001011010 на вход кодирующего устройства — семейство диаграмм (рис. 7).

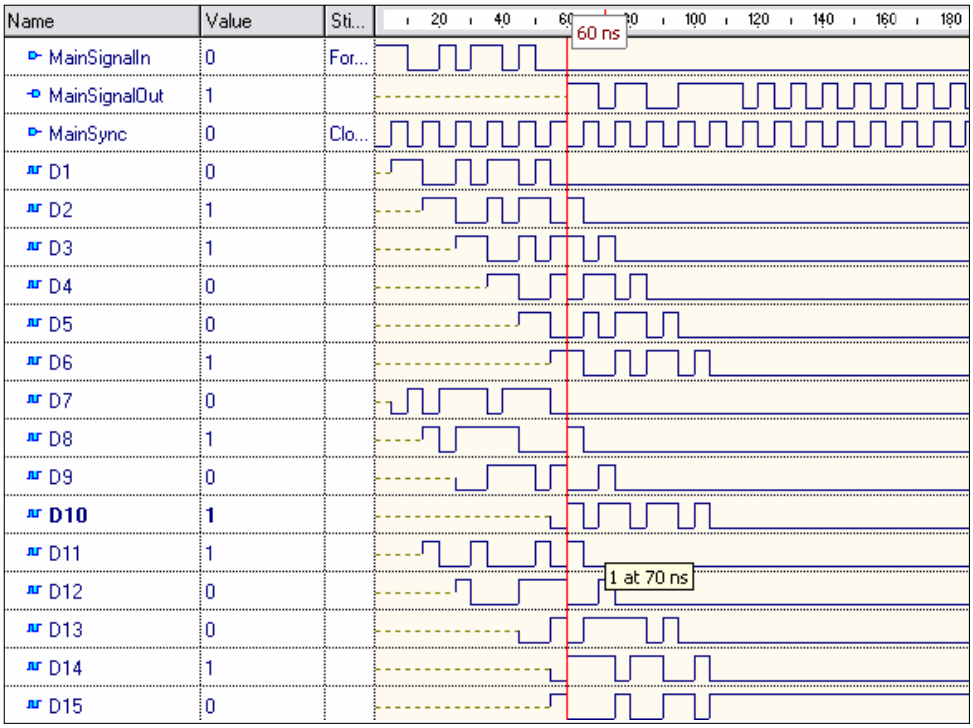

Рис. 7. Процедура кодирования

Следует отметить, что код начинает проходить на выход через 60нс после поступления первого бита на вход кодера. В некоторых случаях для правильной работы может понадобиться подача нулей на вход в течение 70 нс для выхода кодирующего устройства на рабочий режим.

Моделирование кодера Рида–Соломона в MATLAB. Любое конечное поле — векторное пространство, элементы которого можно представить различными способами. Например, в виде последовательностей из n элементов поля. С другой стороны, каждому вектору $(a_0, a_1, ..., a_{n-1})$ можно сопоставить многочлен

$$a_0 + a_1 X + a_2 X^2 + ... + a_{n-1} X^{n-1},$$

называемый *производящей функцией* или *формальным степенным рядом*.

Так как конечное поле F является векторным пространством размерности k, то любой элемент поля F можно представить в виде многочлена степени $k-1$.

Пусть $g(X) \in F[X]$ — некоторый неприводимый многочлен степени m. Отнесем к одному классу все многочлены, которые дают один и тот же

остаток при делении на $g(X)$. Два многочлена $f_1(X)$ и $f_2(X)$ принадлежат одному классу тогда и только тогда, когда $f_1(X) - f_2(X)$ делится на $g(X)$. Полученные таким образом классы называются классами вычетов по модулю $g(X)$, а их множество обозначается $F[g]$. Введем операции сложения и умножения в этом множестве.

$$(f(X)) + (h(X)) = (f(X) + h(X)),$$

$$(f(X))(h(X)) = (f(X)h(X))$$

$$\text{для любых} \quad (f(X)), (h(X)) \in F[g]. \tag{1}$$

Если α — класс вычетов (X), содержащий многочлен X, то многочлен $g(X)$ имеет корень α в поле $F[g]$. Каждый элемент β поля $F[g]$ можно представить в виде

$$\beta = a_{m-1}\alpha^{m-1} + ... + a_1\alpha + a_0, \quad a_i \in F, \quad 1 \le i \le m. \tag{2}$$

Это так называемое *многочленное представление* элементов поля.

Например, необходимо построить поле порядка 2^3, используя неприводимый многочлен $X^3 + X + 1$. Положим $g(X) = X^3 + X + 1$ и рассмотрим восемь классов вычетов по модулю $g(X)$.

$$\{(0), (1), (X), (X+1), (X^2), (X^2+1), (X^2+X), (X^2+X+1)\}. \tag{3}$$

Определим сложение и умножение этих классов вычетов следующим образом:

$$(f(X)) + (h(X)) = (f(X) + h(X)),$$

$$(f(X))(h(X)) = \text{ Остаток от деления } f(X)h(X) \text{ на } g(X). \tag{4}$$

Поскольку примитивный элемент поля p есть 2, то сложение выполняется по модулю 2. Результатом умножения будет остаток от деления их произведения на $g(X)$

$$\frac{X(X^2+X)}{X^3+X+1} = \frac{X^3+X^2}{X^3+X+1} = 1 + \frac{X^2-X-1}{X^3+X+1}. \tag{5}$$

Итак, в данном поле $2 \times 6 = 7$, так как вычитание и сложение по модулю 2 эквивалентны.

Для получения таблицы умножения поля порядка 2^3 можно использовать следующую программу для среды MATLAB [5] :

```
m = 3;
els = gf([0:2^m-1]', m); % создание конечного поля
multabl = gf(zeros(8,8), m);
for j = 1:2^m % получение таблицы умножения
   for i = 1:2^m
      multabl(j,i) = els(j) * els(i);
```

end;

end;

В результате выполнения программы получаем таблицу умножения в конечном поле $GF(8)$

$$\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 3 & 1 & 7 & 5 \\ 0 & 3 & 6 & 5 & 7 & 4 & 1 & 2 \\ 0 & 4 & 3 & 7 & 6 & 2 & 5 & 1 \\ 0 & 5 & 1 & 4 & 2 & 7 & 3 & 6 \\ 0 & 6 & 7 & 1 & 5 & 3 & 2 & 4 \\ 0 & 7 & 5 & 2 & 1 & 6 & 4 & 3 \end{array}.$$

Коды Рида–Соломона. Это циклические коды длиной $q-1$ и порождающим многочленом $g(X)=\prod_{i=1}^{r}(X-\beta^i)$, где β — один из примитивных элементов поля $GF(q)$. Так как для кодов Рида–Соломона минимальное расстояние $d=r+1$, то они являются разделимыми кодами с максимально достижимым расстоянием и используются для построения кодов, исправляющих ошибки.

Например, для рассмотренного выше конечного поля получение кодов Рида–Соломона и их декодирование в среде MATLAB может быть выполнено в соответствии с такой процедурой:

$m=3$; % Number of bits in each symbol (число бит в символе)

$n=7;\ k=3$; % Codeword length and message length (длина кодового слова и число передаваемых символов)

$msg=gf([3\ 7\ 6, m])$; % Represent data using a Galois array (представление данных полем Галуа)

$c1=rsens(msg,n,k)$

$d1=rsdec(c1,n,k)$

В результате получаем элементы кодового слова в массиве $c1$:

$c1=GF(2\text{^}3)$ array. Primitive polynomial (примитивный многочлен) $=$ $=D\text{^}3+D+1$ (11 decimal (десятичное))

Array elements (элементы массива) $=3\ \ 7\ \ 6\ \ 1\ \ 2\ \ 0\ \ 5$

и декодированные данные в массиве $d1$:

$d1=GF(2\text{^}3)$ array. Primitive polynomial $=D\text{^}3+D+1$ (11 decimal)

Array elements $=3\ \ 7\ \ 6$

Определим производящий многочлен:

$\text{rsgp}=\text{rsgenpoly}(7,3)$

$\text{rsgp}=GF(2\text{^}3)$ array. Primitive polynomial (примитивный многочлен) $=$ $=D\text{^}3+D+1$ (11 decimal (десятичное))

Array elements (элементы массива) $=1\ \ 3\ \ 1\ \ 2\ \ 3$

Отметим, что составление блок-схемы кодера для кода Рида–Соломона (рис. 8) в конечном поле $GF(2^3)$ выполняется на основе таблицы умножения и коэффициентов производящего многочлена (см. пример выше).

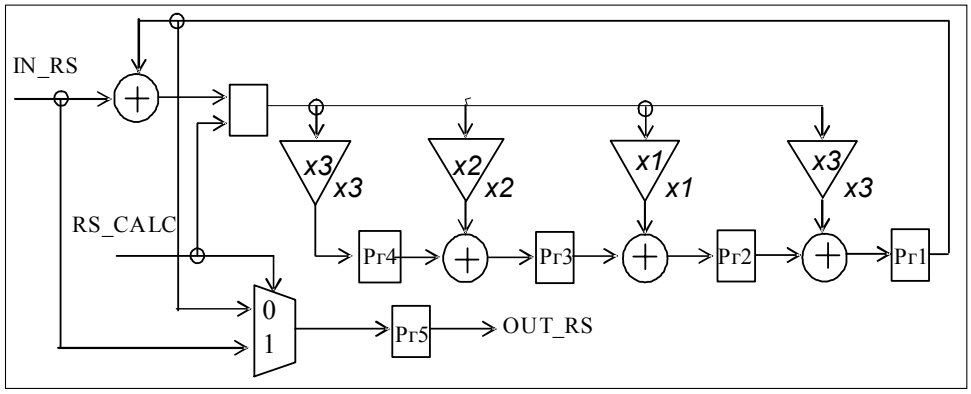

Рис.8. Структурная схема кодера для кода Рида–Соломона

Моделирование кодера для кода Рида–Соломона в среде Active-HDL. Рассмотрим структуру кодера Рида–Соломона для конечного поля $GF(2^3)$ (рис. 9). Сигнал IN_RS — это входная последовательность из k 3-битовых символов, OUT_RS — кодовое слово, состоящее из n символов, из которых k символов входные, а p следующих символа — паритетные (контрольные). Сигнал RS_CALC управляет передачей символов на выходе. При высоком уровне сигнала на выход передается последовательность входных символов, при низком — паритетных. Рг1÷Рг4 — 3-битовые регистры для формирования паритетных символов, умножители на их входах соответствуют коэффициентам производящего многочлена со второго по пятый. Поскольку первый коэффициент всегда равен единице, он учитывается неявно. Сумматоры производят сложение по модулю 2. Рг5 используется для формирования выходной последовательности.

Результаты моделирования работы кодера **RSenc** показаны временными диаграммами на рис. 9.

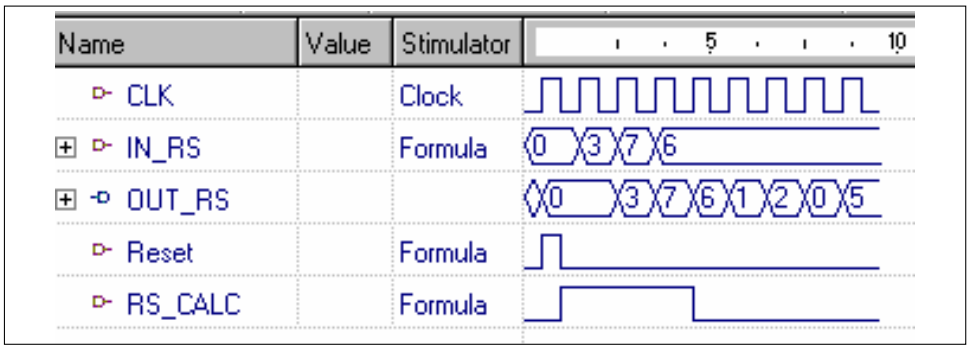

Рис. 9. Диаграммы работы кодера для кода Рида–Соломона

На вход кодера была подана последовательность [3 7 6] , в результате на выходе получено кодовое слово [3 7 6 1 2 0 5], что соответствует решению примера в среде MATLAB.

Турбокодер. Турбокоды относятся к блочным кодам с большим размером блоков ($\sim 10^2 \div 10^3$). Это систематические коды, не являющиеся прозрачными. Вопрос фазовых несоответствий решается с помощью добавления синхромаркера (ASM), необходимого для синхронизации кодового блока.

Турбокодер состоит из двух отдельных кодеров. На его вход приходит фрейм из k бит информации. Отдельные кодеры — это рекурсивные сверточные кодеры с малым числом состояний. Основное свойство турбокодера — наличие перемежителя, перемешивающего биты входной информации перед тем, как подать их на вход второго кодера. Кодовая скорость турбокодов, которую можно определить как отношение длины блока информации к длине кодового блока, в соответствии с рекомендациями CCSDS может быть $1/2$, $1/3$, $1/4$ или $1/6$. Длину блока информации выбираем равной 1784 бита: $1784 = 233x1$ октет, а для указанных выше кодовых скоростей длина соответствующих кодовых блоков будет 3576, 5364, 7152 и 10728 бит.

Перемежитель турбокодов переставляет биты входного блока по постоянному закону. В соответствии с рекомендациями CCSDS перестановки бит в блоке длиной k определяются изменением порядка натурального ряда целых чисел $1,2,...,k$ по следующему алгоритму.

Представим k как $k = k_1 k_2$. Например, $k_1 = 8$, а $k_2 = 223$ и определим перестановочный индекс $\pi(s)$ на основе определенного множества простых чисел $p = [31,37,43,47,53,59,61,67]$.

$$s = 1,2,...k,\ m = \left[\frac{s-1}{2k_2}\right],\ j = \left[\frac{s-1}{2}\right] - ik_2,\ \ t = (19i-1) \bmod \frac{k_1}{2},$$

$$q = i \bmod 8 + 1,\ \ c = (p_q j + 21m) \bmod k_2,\ \ \pi(s) = 2\left(t + c\frac{k_1}{2} + 1\right) - m. \qquad (6)$$

Этот алгоритм реализован в среде MATLAB в виде такой процедуры:

```
% Turbo interleaver
k1 = 8;
k2 = 223;
k = k1 * k2;
p = [31,37,43,47,53,59,61,67];
for s = 1:k;
m = mod((s-1),2);
i = fix((s-1)/(2*k2));
j = fix((s-1)/2) - i*k2;
t = mod((19*i+1),k1/2);
q = mod(t,8) + 1;
c = mod((p(q)*j + 21*m),k2);
pii(s) = 2*(t + c*k1/2 + 1);
end;
```

Результатом выполнения является файл, содержащий индексы переставляемых бит:

4,171,300,467,596,763,892,1059,1188,1355,1484,1651,1780,163,292,459,
588,755,884,1051,1180,1347,1476,1643,1772,155,284,451,580,747,876,
1043,…,1230,1397,1574,1741,134,301,478,645,822,989,1166,1333,1510,
1677,70,237,414,581,758,925,1102,1269,1446,1613.

Полученный массив индексов в дальнейшем используется при построении VHDL-модели перемежителя.

Структурная схема турбокодера, спроектированного в соответствии с рекомендациями CCSDS, приведена на рис. 10. Структура блоков «кодер_а» и «кодер_в» одинакова (рис. 11).

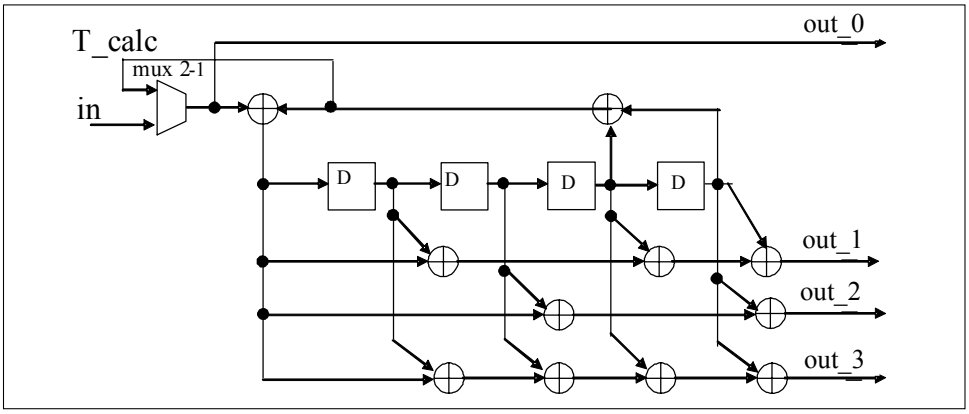

Рис. 10. Структурная схема турбокодера

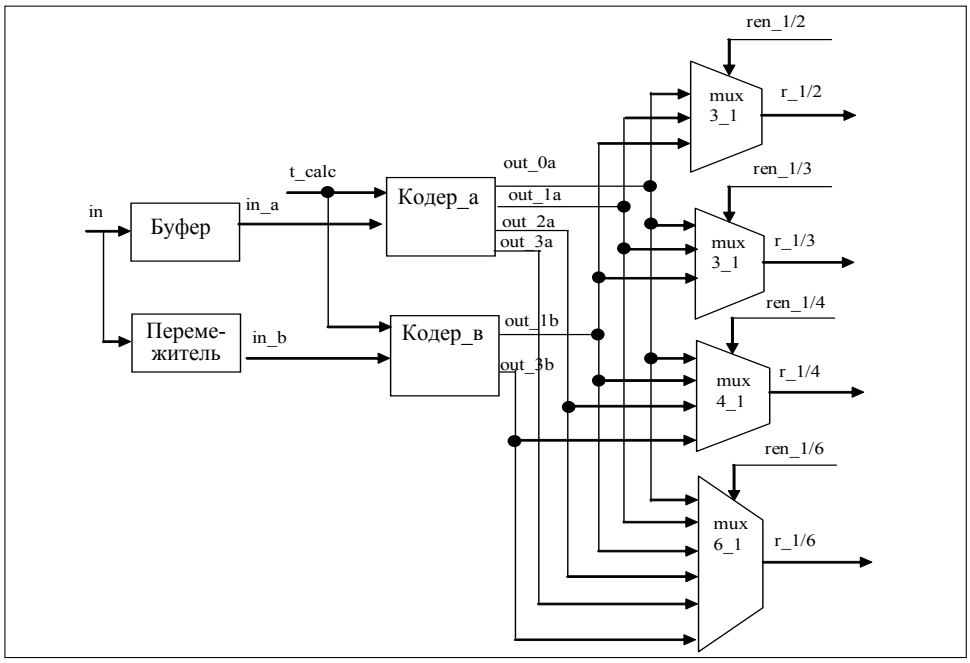

Рис. 11. Структура кодера

VHDL–модель турбокодера. Верхний уровень проекта состоит из испытательного стенда для проверки VHDL–модели турбокодера. Структур-

ная схема испытательного стенда приведена на рис. 12. В блоке Source вырабатывается последовательный код, который затем подается на информационный вход блока turbostr, являющегося VHDL–моделью турбокодера. Назначение сигналов-стимулов следующее:

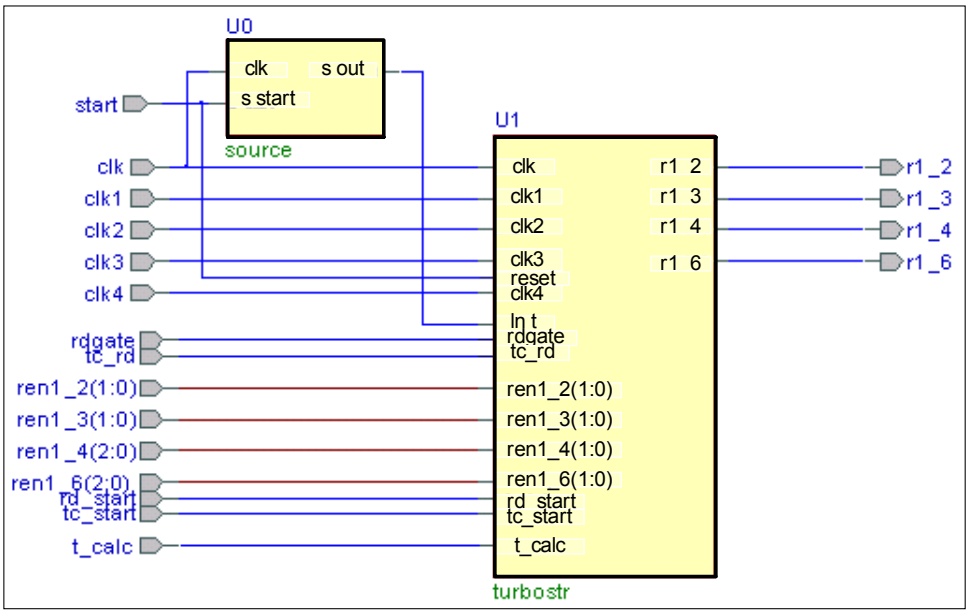

Рис. 12. Испытательный стенд проекта

clk: in STD_LOGIC — синхроимпульсы с частотой соответственно 2; 3; 4; 6 МГц;

clk1; 2, 3, 4: in STD_LOGIC — синхроимпульсы с частотой 2; 3; 4; 6 МГц для формирования кодового блока на кодовой скорости 1/2; 1/3; 1/4;1/6;

rd_start: in STD_LOGIC — стартовый сигнал процесса считывания с буфера входной информации;

rdgate : in STD_LOGIC — сигнал разрешения считывания с буфера входной информации;

start : in STD_LOGIC — сигнал запуска процесса проверки турбокодера и сброса триггеров памяти;

t_calc : in STD_LOGIC — сигнал начала формирования паритетных символов в «хвосте» кодового блока;

tc_rd : in STD_LOGIC — сигнал начала считывания информации с обоих кодеров;

tc_start : in STD_LOGIC — стартовый сигнал для турбокодера;

ren1_2, 3 : in STD_LOGIC_VECTOR (1 downto 0) — переключение входов **mux3_1** для формирования кодового блока на кодовой скорости 1/2; 1/3;

ren1_4 : in STD_LOGIC_VECTOR (2 downto 0) — переключение входов **mux4_1** для формирования кодового блока на кодовой скорости 1/4;

ren1_6 : in STD_LOGIC_VECTOR (2 downto 0) — переключение входов **mux6_1** для формирования кодового блока на кодовой скорости 1/6.

С выходов блока **turbostr** поступает информация:

r1_2, 3, 4, 6 : out STD_LOGIC — кодовый блок на кодовой скорости 1/2; 1/3; 1/4; 1/6.

Результат проверки показан на рис. 13.

Рис. 13. Временные диаграммы проверки турбокодера на кодовой скорости 1/2: *а* — полный временной диапазон моделирования; *б* — моделирование «хвоста» кодового блока (увеличенный масштаб)

Алгоритм наложения «шума». Для гарантии достоверности передачи необходимо каждый бит кодового блока или фрейма передачи через «исключающие ИЛИ» соединить со стандартной псевдослучайной последовательностью.

Если наложение «шума» используется в начале линии (передающая сторона), то его применяют к сигналу кодового блока или фрейма передачи после турбокодирования или R-S-кодирования в зависимости от используемого алгоритма. Если же используется сверточное кодирование, то наложение «шума» нужно производить до кодового блока. В конце линии (принимающая сторона) его используют в качестве блока, выполняющего функцию, обратную наложению «шума» данных после сверточного декодера (в случае использовыания) и кодового блока синхронизатора, но перед R-S или турбодекодером (если они используются).

Следует отметить, что операция, обратная наложению «шума», может производиться одним из способов: 1 — с помощью операции «исключающее ИЛИ» над псевдослучайной последовательностью и полученными битами фрейма передачи или R-S-кодового блока; 2 — инвертированием (или не инвертированием) в зависимости от конфигурации бита псевдорандомизатора выхода демодулятора кодового блока Турбо.

Конфигурация псевдорандомизатора, установленного в начале линии (передающая сторона), показана на рис. 14.

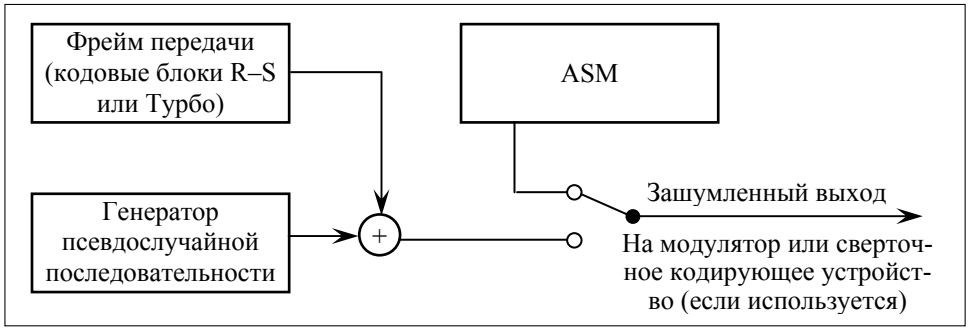

Рис. 14. Конфигурация блока наложения «шума»

Добавленный синхромаркер (ASM) уже оптимально сконфигурирован для целей синхронизации и поэтому используется для синхронизации потока «шума». Рандомизация применяется с начала первого бита фрейма передачи или кодового блока. В начале линии (передающая сторона) кодовый блок или фрейм передачи зашумляются проведением операции «исключающее ИЛИ» над первым битом кодового блока или фрейма передачи с первым битом псевдослучайной последовательности, далее вторым битом кодового блока или фрейма передачи со вторым битом псевдослучайной последовательности и т.д.

В конце линии (принимающая сторона) оригинал кодового блока или фрейм передачи восстанавливаются той же самой псевдослучайной последовательностью. После расположения ASM в полученном потоке данных псевдослучайная последовательность поступает на вход схемы «исключающее ИЛИ» с битами данных сразу же после ASM. Рандомизация выполняется с помощью операции «исключающее ИЛИ» первого бита после ASM с первым ее битом, вторым битом потока данных с ее вторым битом и т. д.

Над псевдослучайной последовательностью и ASM не производится операция «исключающее ИЛИ».

Псевдослучайная последовательность должна быть сгенерирована с использованием следующего полинома:

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1 .$$

Эта последовательность начинается в первом бите кодового блока или фрейма передачи и повторяется после 255 битов, продолжая генерироваться неоднократно до конца фрейма передачи или кодового блока. Генератор последовательности инициализирован к состоянию «все единицы» в начале каждого кодового блока или фрейма передачи.

Первые 40 бит псевдослучайной последовательности от генератора показаны ниже. Крайний левый бит является первым битом последовательности, который будет сложен с первым битом фрейма передачи или кодового блока в схеме «исключающее ИЛИ», второй бит последовательности соответственно со вторым битом кодового блока или фрейма передачи и т. д.

1111 1111 0100 1000 0000 1110 1100 0000 1001 1010... (*)

На рис. 15 приведена структура генератора указанной последовательности.

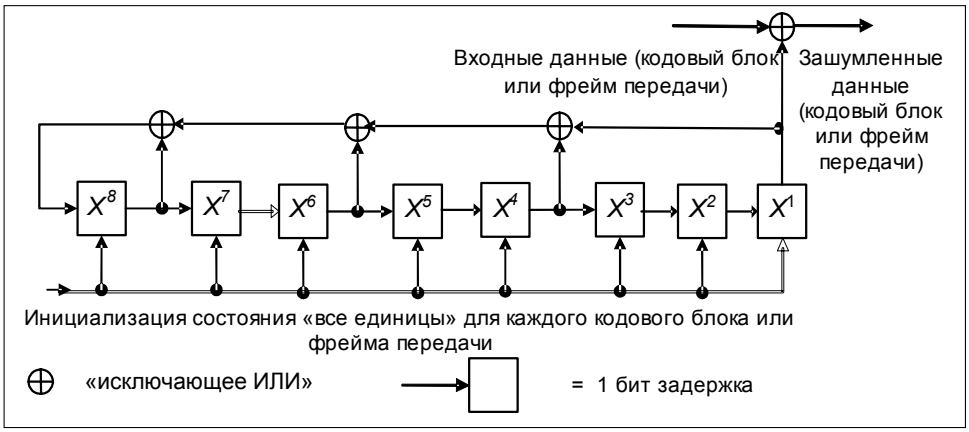

Рис. 15. Структурная схема генератора псевдослучайной последовательности

Моделирование генератора псевдослучайной последовательности в среде Active–HDL. Проект состоит из двух файлов — *BitDelay.vhd* и *Main.vhd*. Первый выполняет задержку каждого бита в каждом блоке задержки. Сигнал *Sync* играет роль синхроимпульса. Следует отметить, что на каждый блок задержки, а также на ключ приходит синхроимпульс с частотой 100 МГц.

Присвоим сигналам имена, для чего используем схему алгоритма. *MainSignalIn* и *MainSignalOut* — входные и выходные сигналы. *MainSync* играет роль синхроимпульса. Сигналы *D1÷D11* распределены так, как показано на рис. 7. Результат моделирования генератора псевдослучайной последовательности показан на рис. 16.

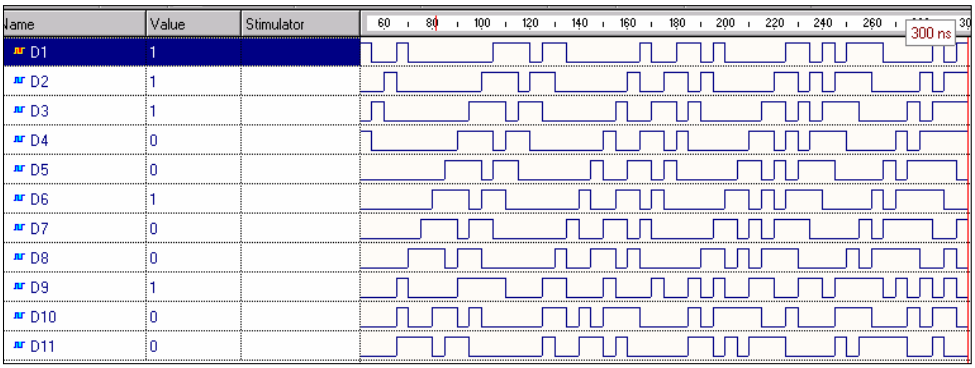

Рис. 16. Результат выполнения программы

ВЫВОДЫ

Приведенные в данной работе методы кодирования информации ТК, структур кодеров и декодеров, рандомизатора формирователя ТК обладают полнотой, достаточной для его моделирования с помощью VHDL–проекта.

В состав формирователя ТК входят блоки сверточного кодирования–декодирования, рандомизатора и блок добавления синхромаркера (ASM) структурной схемы, составленной в соответствии с рекомендациями CCSDS.

В дальнейшем предполагается проверить правильность функционирования формирователя ТК на тестовых стендах (test bench). Затем спроектировать структуру для загрузки проекта в программируемые пользователем логические интегральные схемы ПЛИС XCV 300E–PQ240 [7]. После отладки добавить блоки кодирования–декодирования по методам Рида–Соломона и Турбо, проверив тем самым возможности реконфигурирования формирователя ТК.

ЛИТЕРАТУРА

1. *Олифер В.Г., Олифер Н.А.* Компьютерные сети. Принципы, технологии, протоколы. — СПб.: Питер, 1999. — 672 с.
2. *Бортовые* цифровые вычислительные машины и системы: Учеб. пособие для вузов / Под ред. В.И. Матова. — М.: Высш. шк., 1988. — 216 с.
3. *Фитч В.* Применение микропроцессоров в системах управления. — М.: Мир, 1994. — 463 с.
4. http://www.ccsds.org/
5. *Потемкин В.Г.* Система MatLab: Справочное пособие. — М.: Диалог-МИФИ, 1998. — 350 с.
6. *Программно-управляемый* обмен данными в системах реального времени / В.П. Зинченко, С.В. Зинченко, Ф.Н. Горин, Н.С. Броварская // Технології створення перспективних комп'ютерних засобів та систем з використанням новітньої елементної бази: Сб. науч. тр. — Київ: Ін-т кібернетики ім. В.М. Глушкова НАНУ, 2000. — С. 55 – 60.
7. *David Van den Bout*. The Practical Xilinx® Designer Lab Book, Vertion 1.5. Prentice Hall, Upper Saddle River. — New Jersey, 07458, 1999. — 450 p.

Поступила 14.02.2005