Санкт-Петербургский государственный политехнический университет
Кафедра компьютерных систем и программных технологий

# Надежность систем и устройств

Моисеев Михаил Юрьевич

Лекция №8

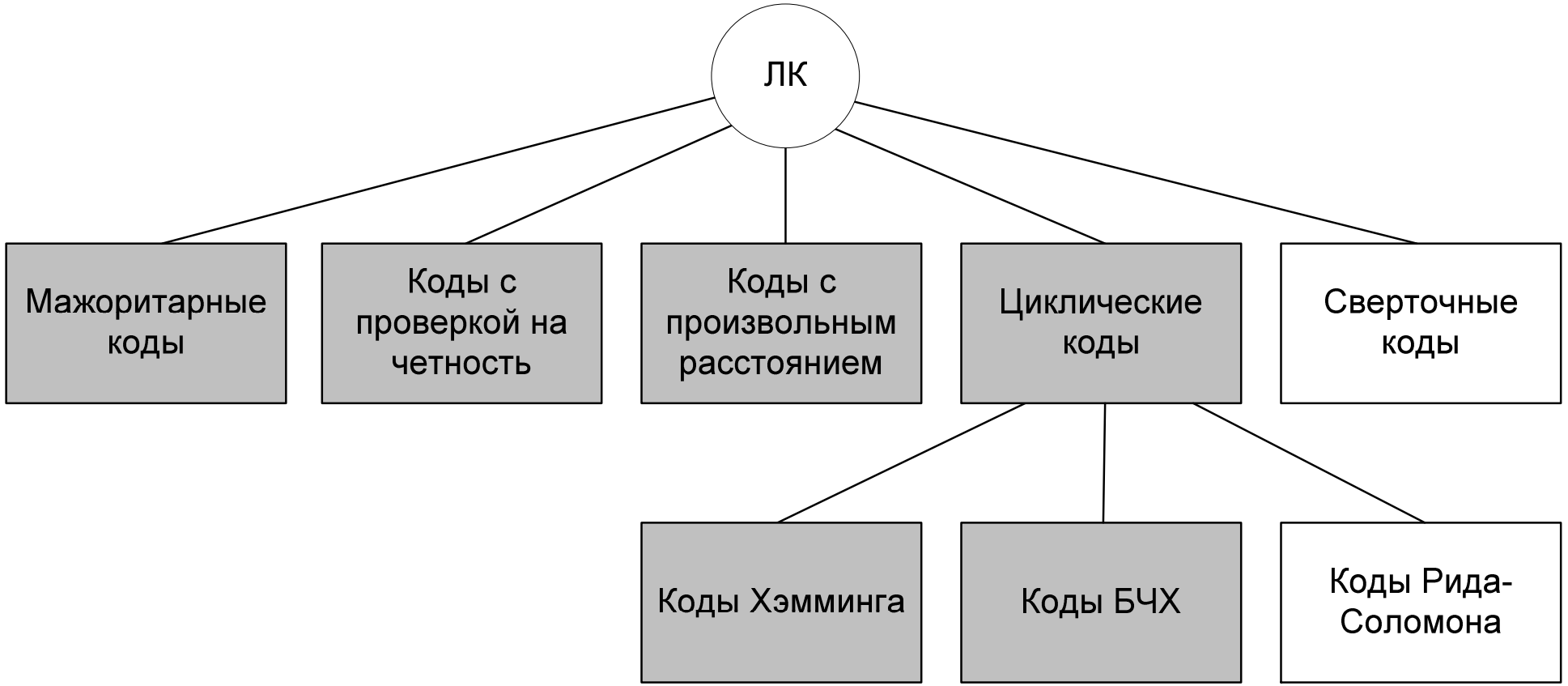
## Коды Рида-Соломона
## Сверточные коды

2011

# Повторение предыдущего материала

- Построить код БЧХ с **k=50** и **t=2**, определить параметры кода, определить скорость кода и вероятность неправильного исправления ошибки, при $p=10^{-2}$

# Рассматриваемые классы кодов



# Коды Рида-Соломона

- Коды Рида-Соломона – подмножество кодов БЧХ с коэффициентами $i(x), c(x)$ из $GF(2^m)$
- Информационные и кодовые вектора являются недвоичными
  - при m=3 – вектора из восьмеричных символов
  - при m=4 – вектора из шестнадцатеричных символов
- Элементы из $GF(2^m)$ могут быть представлены как наборы двоичных символов, длина кода в двоичном представлении равна $n \times m$
- Повышение эффективности коды с ростом длины

# Определение кода Рида-Соломона

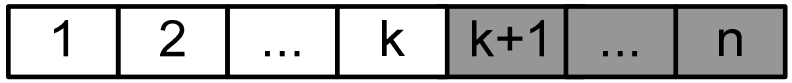
- Код, состоящий из всех многочленов с коэффициентами из поля $GF(q^m)$, для которых элементы $\beta^{j_0}, \beta^{j_0+1}, \beta^{j_0+2}, \ldots, \beta^{j_0+2t-1}$ являются корнями, называется кодом Рида-Соломона
- Код Рида-Соломона исправляет все ошибки кратности **t** и меньше
- Для синтеза кода Рида-Соломона необходимо построить $g(X)$, для которого все указанные элементы являются корнями $c(X) = i(X) \cdot g(X)$
- Построение порождающего многочлена $g(x) = \prod_{i=0}^{2t-1} \left(x - \beta^{j_0+i}\right)$

# Построение кодов Рида-Соломона

- Будем использовать $j_0 = 1$ (**$j_0$** влияет на сложность кодека)

$$g(x) = (x - \beta^1)(x - \beta^2)\ldots(x - \beta^{2t})$$

- Длина кода **$n = 2^m - 1$**
- Степень $g(x)$ равна **$2t \Rightarrow n - k = 2t$**
- Кодовое расстояние **$d = n - k + 1$** (**$2t - 1 = d$**)
- Коды Рида-Соломона являются кодами с максимальным кодовым расстоянием (Почему?)



# Пример построения кода Рида-Соломона

- Построим код Рида-Соломона с **t = 2** в $GF(2^3)$ $p(x) = x^3 + x + 1$

$$\beta^0 = 001 \equiv 1$$
$$\beta^1 = 010 \equiv X^1$$
$$\beta^2 = 100 \equiv X^2$$
$$\beta^3 = 011 \equiv X + 1$$
$$\beta^4 = 110 \equiv X^2 + X$$
$$\beta^5 = 111 \equiv X^2 + X + 1$$
$$\beta^6 = 101 \equiv X^2 + 1$$
$$\beta^7 = 001 \equiv 1$$

$$g(x) = (x - \beta)(x - \beta^2)(x - \beta^3)(x - \beta^4) =$$
$$= x^4 + \beta^3 x^3 + x^2 + \beta\, x + \beta^3$$

- **n = 7**, **n – k = 4** $\Rightarrow$ **(7,3)**-код
- $i =$ (001, 011, 101)

$$i(x) = x^2 + \beta^3 X + \beta^6$$

$$c(x) = i(x) \cdot g(x) =$$
$$x^6 + \beta^6 x^5 + \beta\ x^4 + \beta^3 x^3 + \beta^6 x^2 + \beta^2 x + \beta^2$$

- $c =$ (001, 101, 010, 011, 101, 100, 100)

# Задача 8

- Построить код Рида-Соломона с **t = 1** в $GF(2^4)$, выполнить кодирование вектора (000…000101100010)

$$\beta^0 = 0001 \equiv X^0 \qquad \beta^5 = 0110 \equiv X^2 + X \qquad \beta^{10} = 0111 \equiv X^2 + X + 1$$
$$\beta^1 = 0010 \equiv X^1 \qquad \beta^6 = 1100 \equiv X^3 + X^2 \qquad \beta^{11} = 1110 \equiv X^3 + X^2 + X$$
$$\beta^2 = 0100 \equiv X^2 \qquad \beta^7 = 1011 \equiv X^3 + X + 1 \qquad \beta^{12} = 1111 \equiv X^3 + X^2 + X + 1$$
$$\beta^3 = 1000 \equiv X^3 \qquad \beta^8 = 0101 \equiv X^2 + 1 \qquad \beta^{13} = 1101 \equiv X^3 + X^2 + 1$$
$$\beta^4 = 0011 \equiv X + 1 \qquad \beta^9 = 1010 \equiv X^3 + X \qquad \beta^{14} = 1001 \equiv X^3 + 1$$
$$\beta^{15} = 0010 \equiv X$$

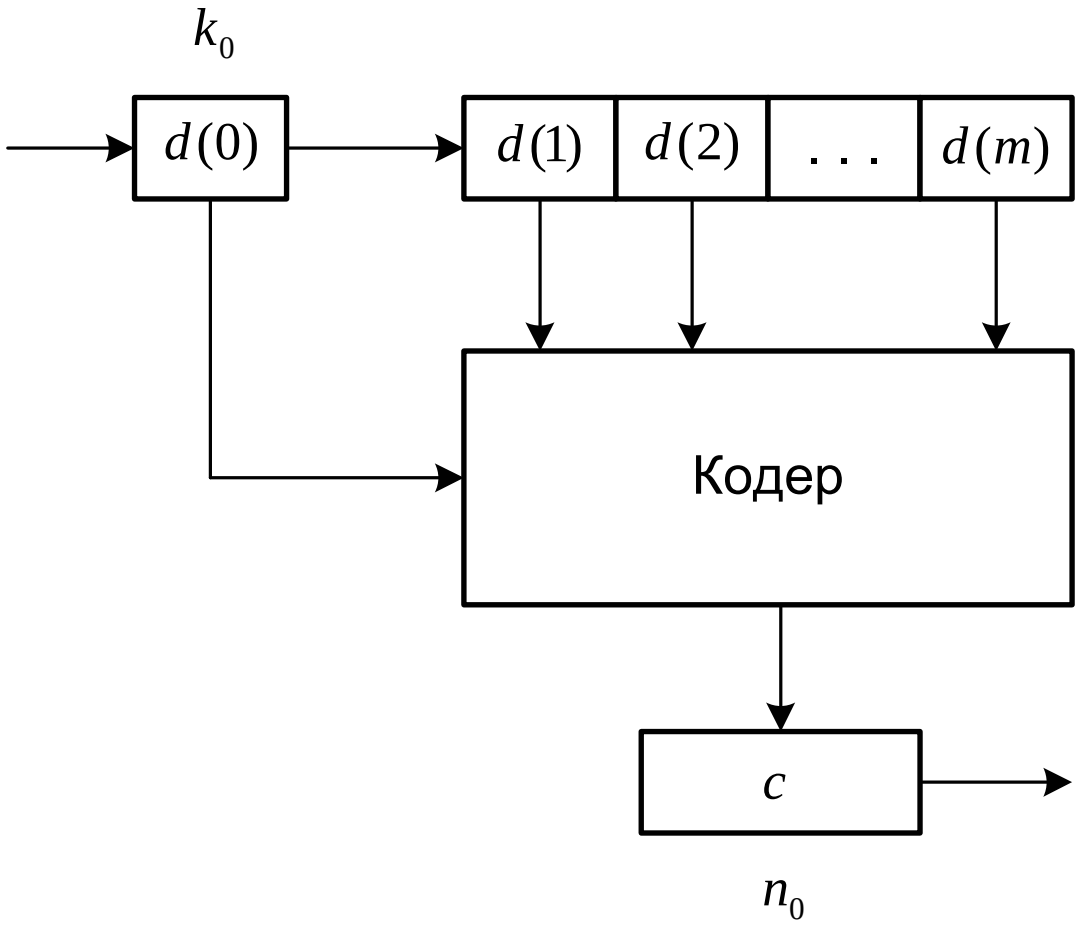
# Декодирование кодов Рида-Соломона

- Могут использоваться алгоритмы декодирования линейных кодов
- Могут использоваться алгоритмы декодирования кодов БЧХ

# Сверточные коды

- Для блоковых кодов информационная последовательность разбивается на блоки длиной **k** символов, блоки кодируются независимо друг от друга
- Для сверточных кодов информационная последовательность разбивается на кадры длиной в $\mathbf{k_0}$ символов. Каждый кадр кодируются с учетом **m** предыдущих кадров

# Общая схема кодера сверточного кода

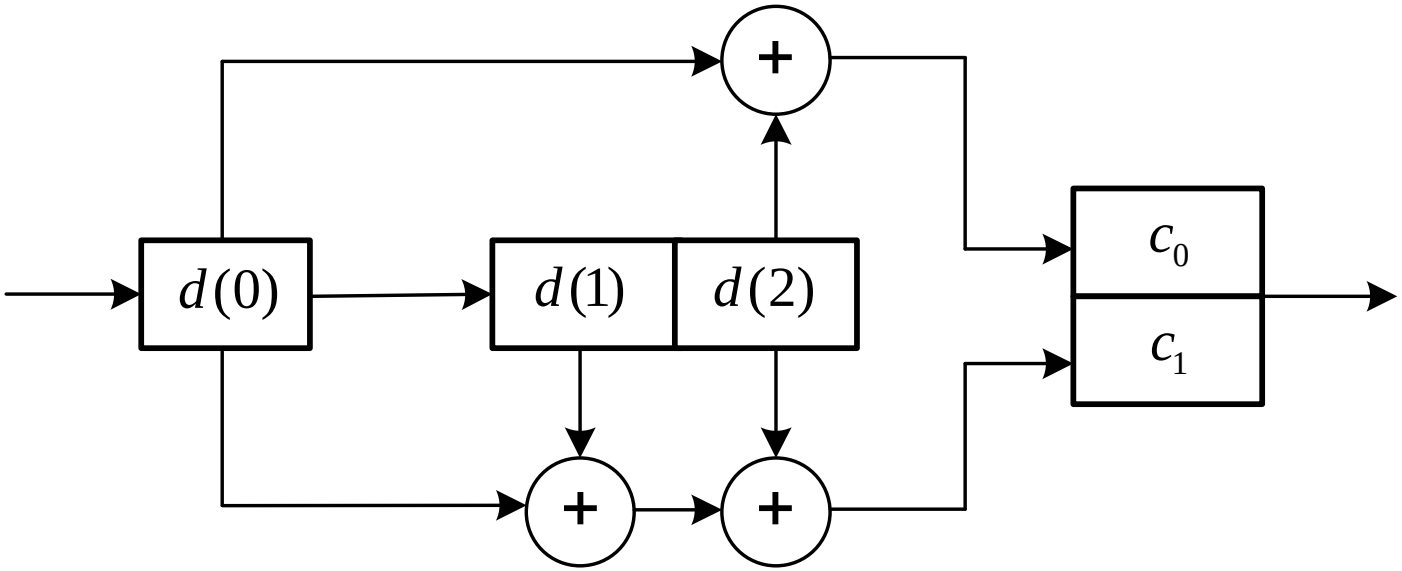


# Свойства сверточных кодов

- Сверточные коды обладают свойствами линейности и постоянства во времени
- Длина кодового ограничения – количество кадров, которые учитываются при кодировании **m**, может быть бесконечной
- Информационная длина $k = k_0(m+1)$
- Кодовая длина $n = n_0(m+1) = k(n_0 / k_0)$
- Сверточный код может задаваться схемой кодера

# Пример кодера сверточного кода

- Рассмотрим кодер сверточного **(6,3)**-кода
- Длина информационного кадра $\mathbf{k_0 = 1}$, длина кодового кадра $\mathbf{n_0 = 2}$



# Задача 9

- Какой сверточный код описывается следующим кодером **$k_0 = 2, n_0 = 3, m = 1$**, нарисуйте схему кодера этого кода если известно что $c_0=d_0(0)+d_1(0)$, $c_1=d_0(0)+d_1(1)$, $c_2=d_1(0)+d_1(1)$

# Описание сверточных кодов с помощью многочленов

- Кодовый кадр зависит определяется **(m+1)** информационными кадрами
- Любой символ кодового кадра можно получить

$$c_j = \sum_{i=0}^{k_0} \sum_{l=1}^{(m+1)} a_{lij} \cdot d_i(l)$$

где $\mathbf{c_j}$ – кодовый символ, $\mathbf{d_i(l)}$ – последовательность информационных символов, $a_{lij}$- некоторые постоянные коэффициенты

# Описание сверточных кодов с помощью многочленов

- Можно записать операцию кодирования с помощью многочленов

$$c_j = g_j(x) \cdot d(x)$$

- В многочлене **d(x)** коэффициенты перед разными степенями **x** соответствуют разным разрядам и разным задержкам $d_i(l)$
- В информационном кадре $\mathbf{k_0}$ символов, многочлен $\mathbf{g_j(x)}$ можно представить как сумму $\mathbf{k_0}$ многочленов ($\mathbf{g_{ij}(x)}$), каждый степени не большей чем **m**
- $\mathbf{g_{ij}(x)}$ – определяет **j**-й разряд кодового кадра, по **i**-м разрядам всех информационных кадров

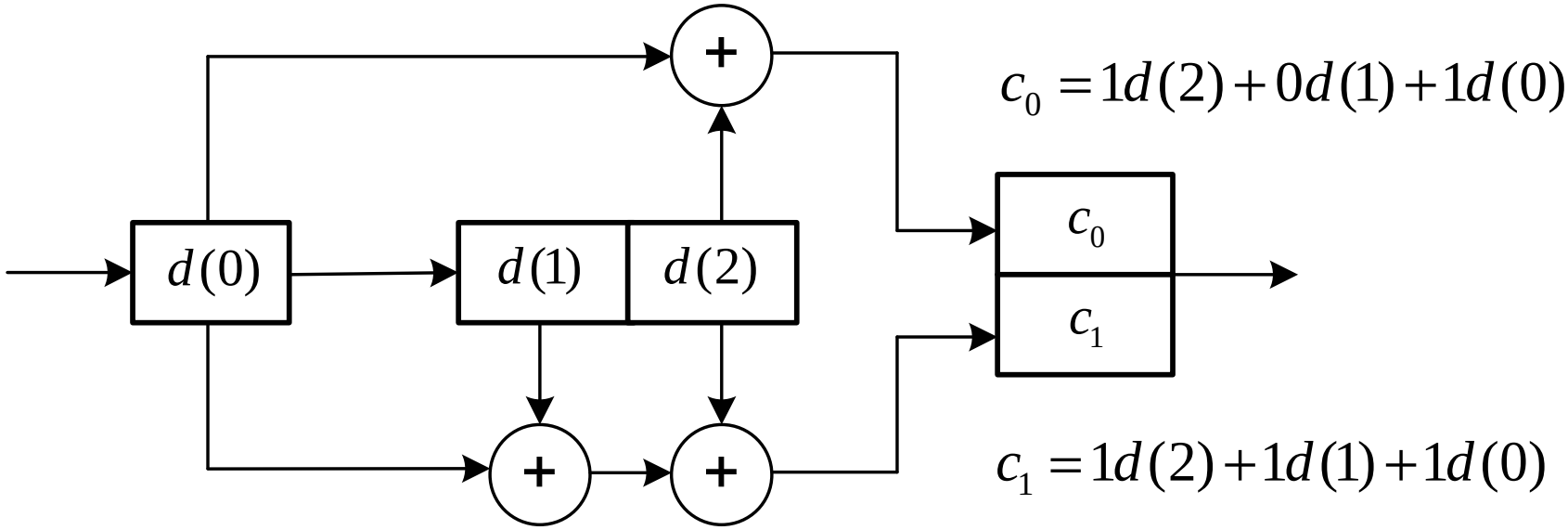
# Матрица многочленов сверточного кода

- Многочлены $\mathbf{g_{ij}(x)}$ называются порождающими многочленами сверточного кода
- Число порождающих многочленов равно $\mathbf{n_0 * k_0}$, некоторые многочлены могут быть одинаковыми
- Матрица порождающих многочленов

$$G(x)_{[k_0 \times n_0]} = [g_{ij}(x)]$$

# Пример кодера сверточного кода

- Представим кодер сверточного **(6,3)**-кода с помощью матрицы порождающих многочленов




$$G_{[1\times 2]}(x) = [x^2 + 1 \quad x^2 + x + 1]$$

# Задача 10

- Записать многочлены по схеме кодера и наоборот построить схему по заданным многочленам

# Корректирующие свойства сверточных кодов

- Минимальное расстояние Хэмминга для начальных последовательностей длиной $l$ - кадров из всех пар кодовых слов, отличающихся начальным кадром, называется $l$-ым минимальным расстоянием и обозначается $d_l^*$
- Последовательность $d_1^*, d_2^*, d_3^*, \ldots d_\infty^*$ называется дистанционным профилем сверточного кода
- Если в первых $l$ кадрах сверточного кода произошло **t** ошибок и $2t+1 \le d_l^*$ то ошибки в первом кадре могут быть исправлены
- Свободным расстоянием сверточного кода называется $d_\infty = \max_l d_l^*$

# Корректирующие свойства сверточных кодов

- Свободной длиной сверточного кода называется наименьшее количество кадров, при котором достигается свободное расстояние сверточного кода $d_1^*, d_2^*, d_3^*, \ldots d_\infty^*$
- Шириной окна декодирования называется число символов, которые используются декодером
- Чем больше ширина окна декодирования, тем больше ошибок может быть исправлено, однако в итоге происходит насыщение
- Ширина окна декодирования от **n** до нескольких **n**

# Синтез сверточных кодов по таблицам

- По требуемым значениям скорости и помехоустойчивости выбирается оптимальный код из таблицы
- Требуется построить код со скоростью 1/3 исправляющий до 4-х ошибок на свободной длине
  - выберем подходящий код с минимальной длиной – **(12,4)**
  - выбираем для этого кода из таблицы все числа 1101, 1011, 1111 и записываем их в виде матрицы многочленов

$$G(x) = [x^3 + x^2 + 1; \quad x^3 + x + 1; \quad x^3 + x^2 + x + 1]$$

  - определяем параметры кода $\mathbf{k_0 = 1}$, $\mathbf{n_0 = 3}$, $\mathbf{m = 3}$

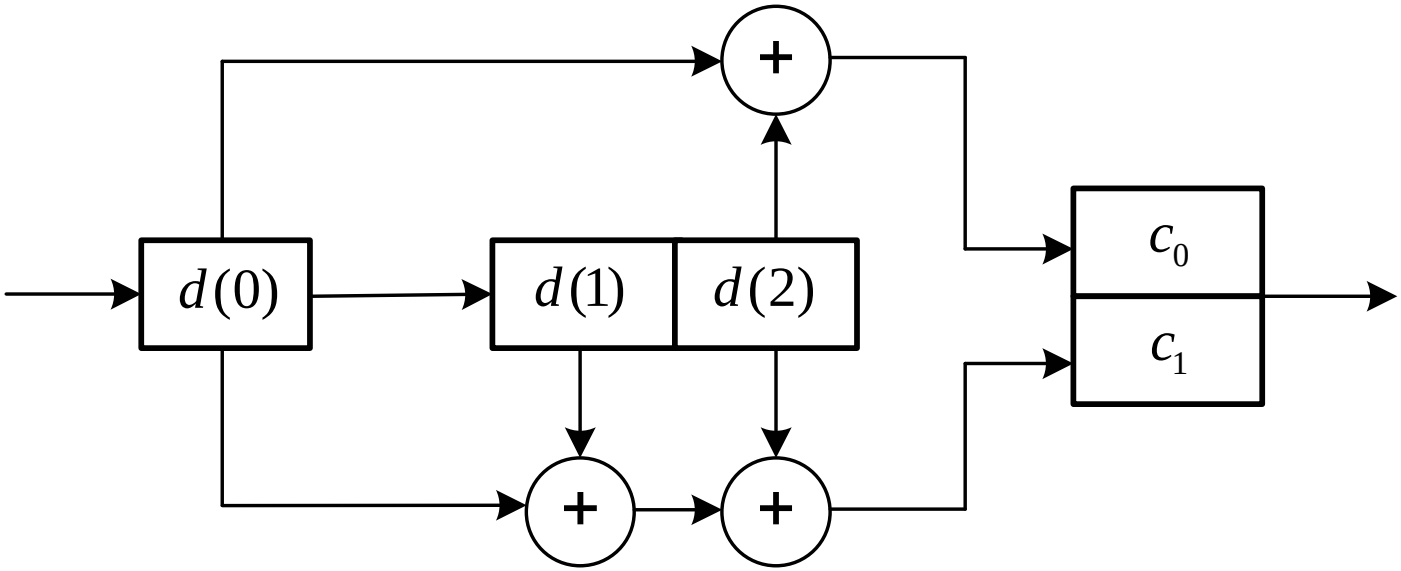
# Задача 11

- Построить код со скоростью 1/2 и t = 3, в виде матрицы **G** и схемы кодера, произвести кодирование информационной последовательности 01010

# Описание сверточного кода с помощью решетки

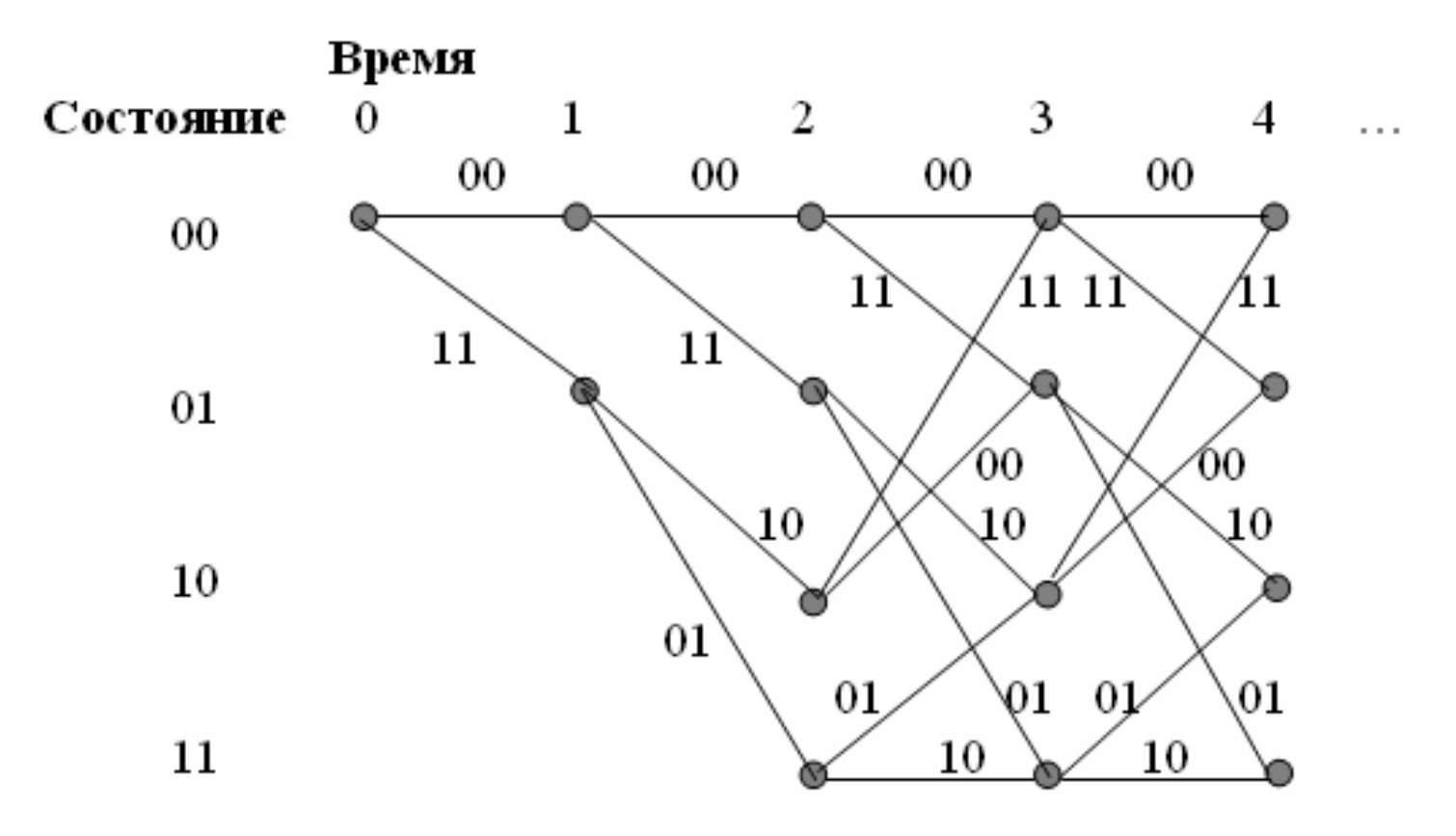
- Сверточные коды можно описывать с помощью решетки
- Узлы в каждом столбце решетки представляют возможные состояний кодера. Каждый следующий столбец представляет собой состояния в следующий момент времени
- При поступлении на вход кодера нового информационного кадра, состояние кодера изменяется, что соответствует ребру перехода к следующему узлу
- Количество ребер исходящих из каждого узла равно числу различных информационных кадров. Маркировка ребер определяет кодовые кадры соответствующие этим ребрам
- Получаемая кодовая последовательность – это путь  по решетке, который определяется информационной последовательностью. Множество всех возможных путей является множеством всех кодовых векторов

# Кодер (6,3)-кода



# Описание сверточного кода с помощью решетки

- **(6,3)**-код, количество ребер – 2, количество состояний кодера – 4
- Верхнее ребро соответствует входному кадру 0, нижнее ребро - 1




$$0110101 \rightarrow 00.11.01.01.00.10.00$$

# Декодирование сверточных кодов

- Сверточный код с $\mathbf{k_0 = 1}$, порождающие многочлены которого удовлетворяют условию $НОД[g_1(x), \ldots, g_{n_0}(x)] = 1$ называется некатастрофическим сверточным кодом
- В противном случае сверточный код называется катастрофическим
- Если ошибка декодирования в одном из кадров приводит к появлению бесконечного числа ошибок в последующих кадрах, то говорят что происходит распространение ошибок
- Для некатастрофического кода распространение ошибок может быть устранено выбором алгоритма декодирования

# Декодирование сверточных кодов

- Для декодирования сверточных кодов достаточно определить декодирование первого кадра
- Если первый кадр декодирован и в нем исправлены ошибки, то влияние информационных символов этого кадра на следующие кадры можно учесть и исключить
- После этого задача декодирования второго кадра аналогична задаче декодирования первого кадра и так далее
- Рассмотрим два алгоритма декодирования с исправлением ошибок
  - алгоритм Виттерби
  - алгоритм последовательного декодирования

# Алгоритм Виттерби

- В алгоритме Виттерби декодер строит множество возможных кодовых последовательностей (путей) на решетке
- При добавлении очередного кадра декодер строит наиболее правдоподобные пути из узлов первого рассматриваемого кадра к каждому из узлов нового кадра (пути с наименьшим расстоянием Хэмминга от принятого вектора)
- Если все построенные пути начинаются в первом кадре все в одном узле, то этот узел  считается верным и производится декодирование первого кадра, иначе декодер отказывается от декодирования
- Декодер может принять неправильное решение и несколько кадров будут декодированы ошибочно, но для некатастрофического кода декодер через некоторое время обнаружит это

# Алгоритм последовательного декодирования

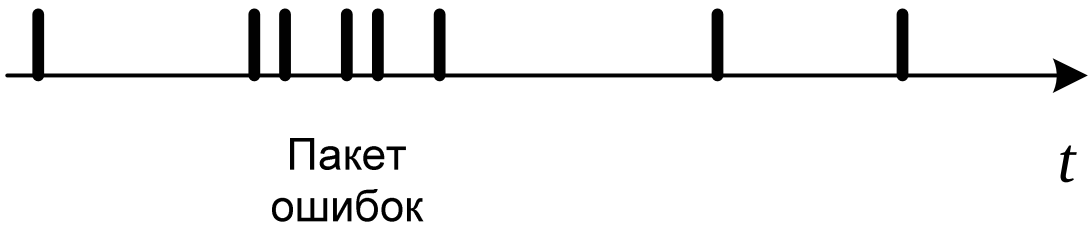
- Строится единственный путь, при приеме первого кадра выносится предварительное решение и проводится декодирование
- Если декодер, начиная с некоторого кадра, пошел по ошибочному пути, то начинает появляться слишком много ошибок, которые возникают не в КС, а в самом декодере
- Обнаружив эту ситуацию, декодер возвращается на некоторое число кадров назад и повторяет попытку, выбирая другой путь
- Правило определения неверного пути

$$t(l) = p'n_0 l - d(l)$$

- $p'$ верхняя граница вероятности ошибки на символ $p < p' < 0.5$, $d(l)$ – расстояние Хэмминга между построенным путем и принятым вектором
- Значение $t(l)$ будет положительным и возрастать пока декодер следует по правильному пути

# Пакеты ошибок

- Пакетом ошибок называется последовательность, первый и последний символы которой ненулевые
- Предполагается, что внутри пакета ошибок вероятность ошибки достаточно высока и больше чем вероятность независимых ошибок
- Пакеты ошибок приходят не часто и можно считать, что одновременно в декодере может находиться только один пакет



# Коды, исправляющие пакеты ошибок

- Любой сверточный код, исправляющий ошибки кратности **t**, будет исправлять и пакеты ошибок длины **t**
- Для исправления более длинных пакетов можно построить код с перемежением символов нескольких кодов
- Используя перемежение символов **j** одинаковых сверточных **(n,k,t)** кодов, можно получить **(jn,jk)** -код, исправляющий пакеты ошибок длины **jt**
- Если многочлен $g(x)$ – порождающий многочлен исходного кода, то многочлен $g(x^j)$ – порождающий многочлен кода перемежения.
- Полученный код будет также исправлять и множество других комбинаций независимых ошибок

# Пример построения код с перемежением символов

- Построим код с перемежением, используя 3 сверточных **(12,4)**-кода исправляющего **4** ошибки
- Получаем **(36,12)**-код, исправляющий пакеты ошибок длины **12**
- Порождающие многочлены исходного кода:

$$g_1(x) = x^3 + x^2 + 1 \quad g_2(x) = x^3 + x + 1 \quad g_3(x) = x^3 + x^2 + x + 1$$

- Порождающие многочлены полученного кода:

$$g_1(x) = x^9 + x^6 + 1 \quad g_2(x) = x^9 + x^3 + 1 \quad g_3(x) = x^9 + x^6 + x^3 + 1$$

# Коды Ивадаре

- Для любых положительных λ и $n_0$ существует $((m+1)n_0,(m+1)(n_0-1))$ код, исправляющий пакеты ошибок длиной $\lambda n_0$, где

$$m=(\lambda+1)(2n_0-1)-2$$

$$G=\begin{vmatrix} 1 & & & g_1(x) \\ & 1 & & g_2(x) \\ & & \cdots & \cdots \\ & & 1 & g_{(n_0-1)}(x) \end{vmatrix}$$

$$g_i(x)=x^{(\lambda+1)(2n_0-i)+i-3}+x^{(\lambda+1)(n_0-i)-1}$$

# Пример построения кода Ивадаре

- Построим код Ивадаре для λ = 2, n0 = 3
- **m + 1 = 14, (42,28)**-код, исправляющий пакеты длины 6

$$G = \begin{vmatrix} 1 & 0 & g_1(x) \\ 0 & 1 & g_2(x) \end{vmatrix}$$

$$g_1(x) = x^{13} + x^5 \qquad g_2(x) = x^{11} + x^2$$

# Вопросы

- Как определяются коды Рида-Соломона? Чем эти коды отличаются от кодов БЧХ?
- Что такое сверточные коды?
- Перечислите способы определения сверточного кода.
- Как определяется кодовое расстояние для сверточных кодов?
- Какие алгоритмы декодирования используются для сверточных кодов?
- Что такое пакет ошибок?
- Какие коды исправляют пакеты ошибок?