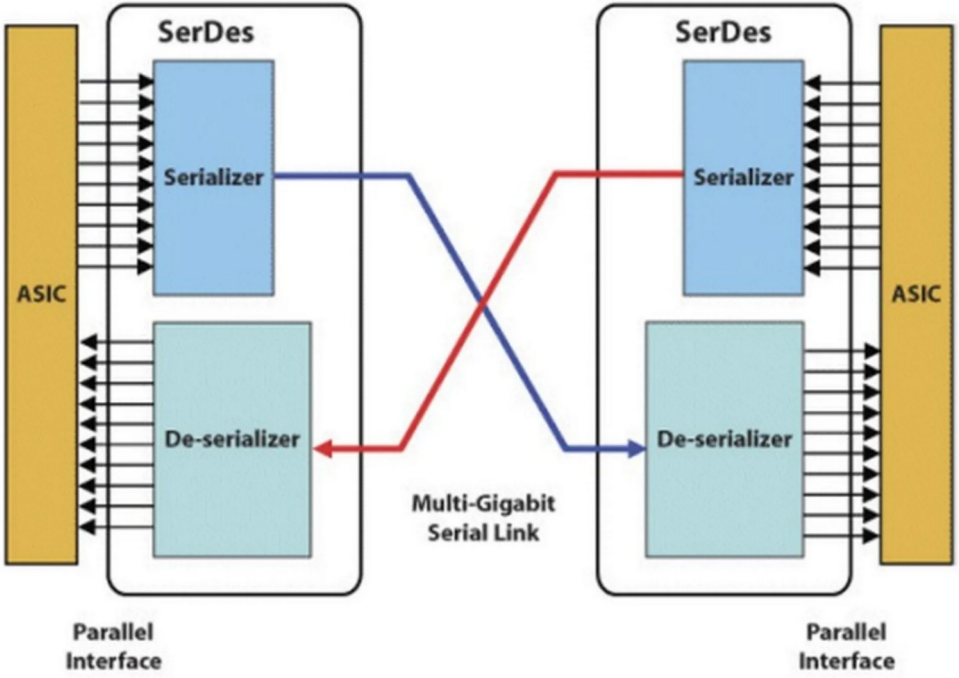
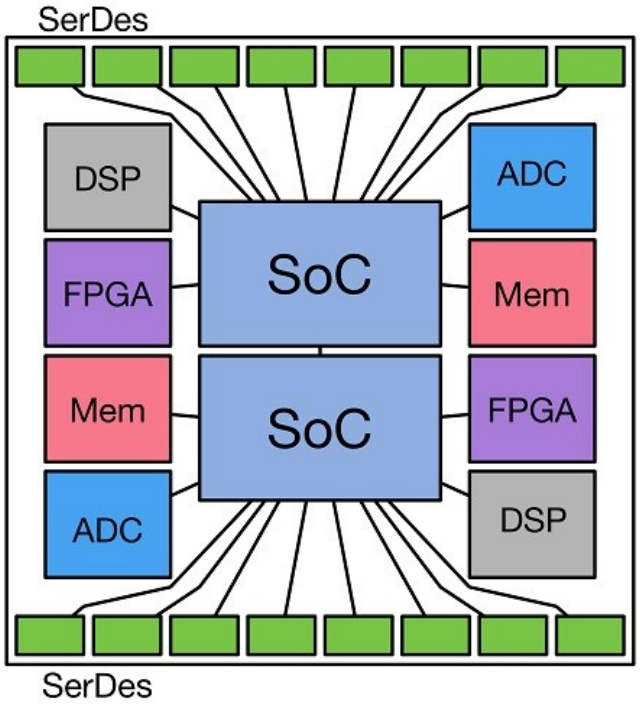
3.1 Сериалайзер / десериалайзер

Проектирование СБИС

Связь между двумя ИС

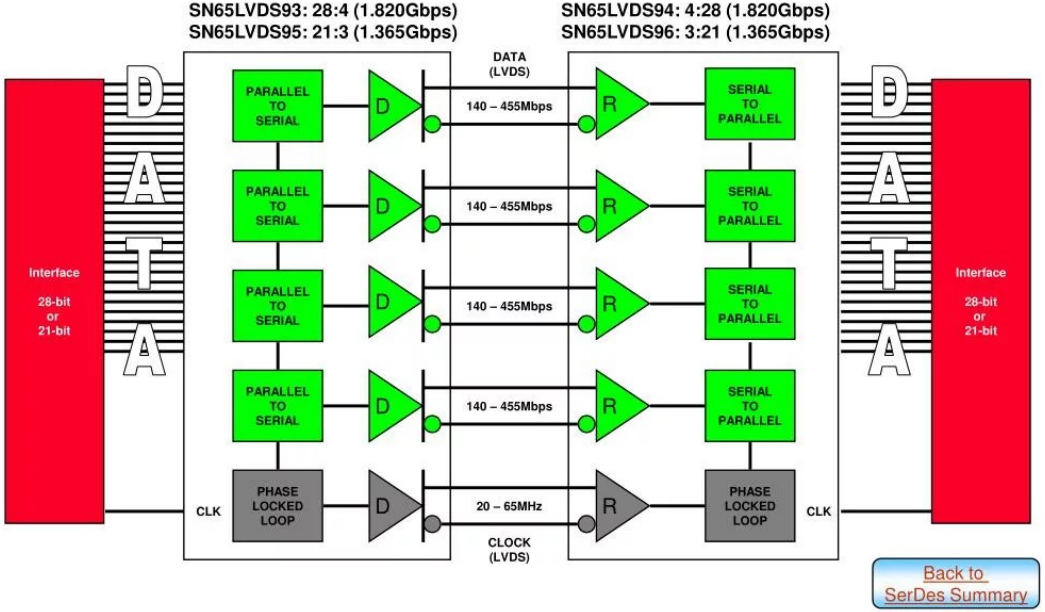




SerDes

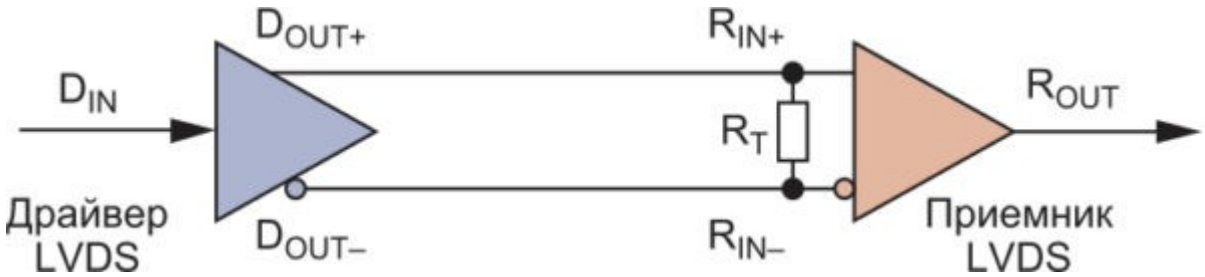
LVDS SERDES Transmitters & Receivers

21:3 / 28:4 and 4:28 and 3:21

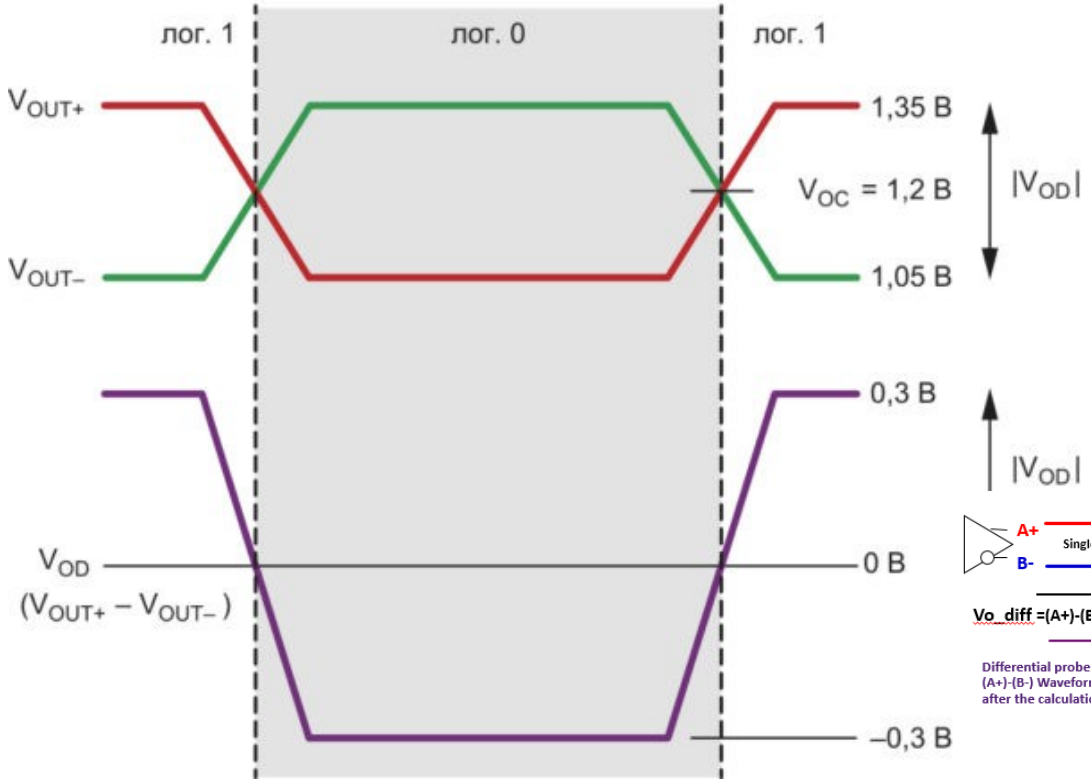


Back to SerDes Summary

LVDS (соединение точка — точка)



Уровни выходных сигналов LVDS



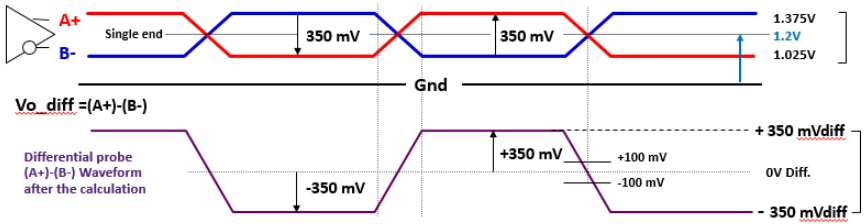
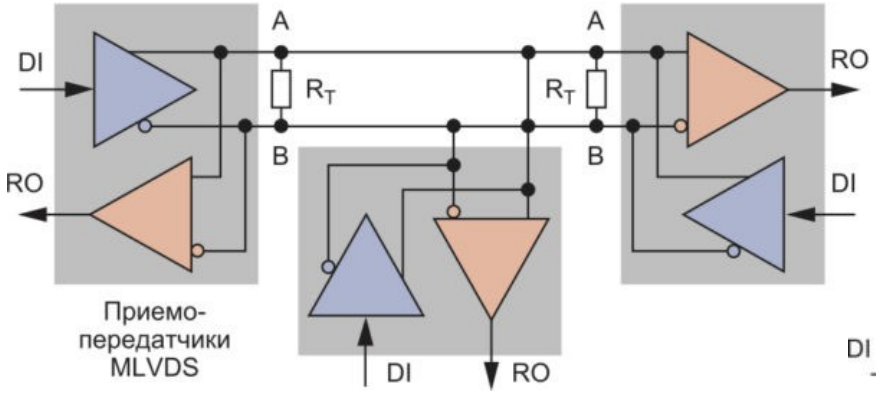


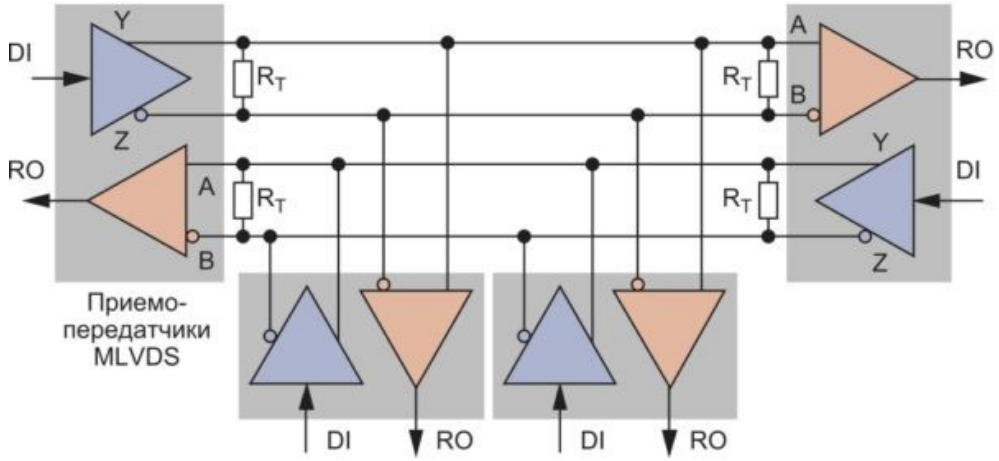
Fig. 2: LVDS differential signals (single end and differential)

M-LVDS (многоточечное соединение)

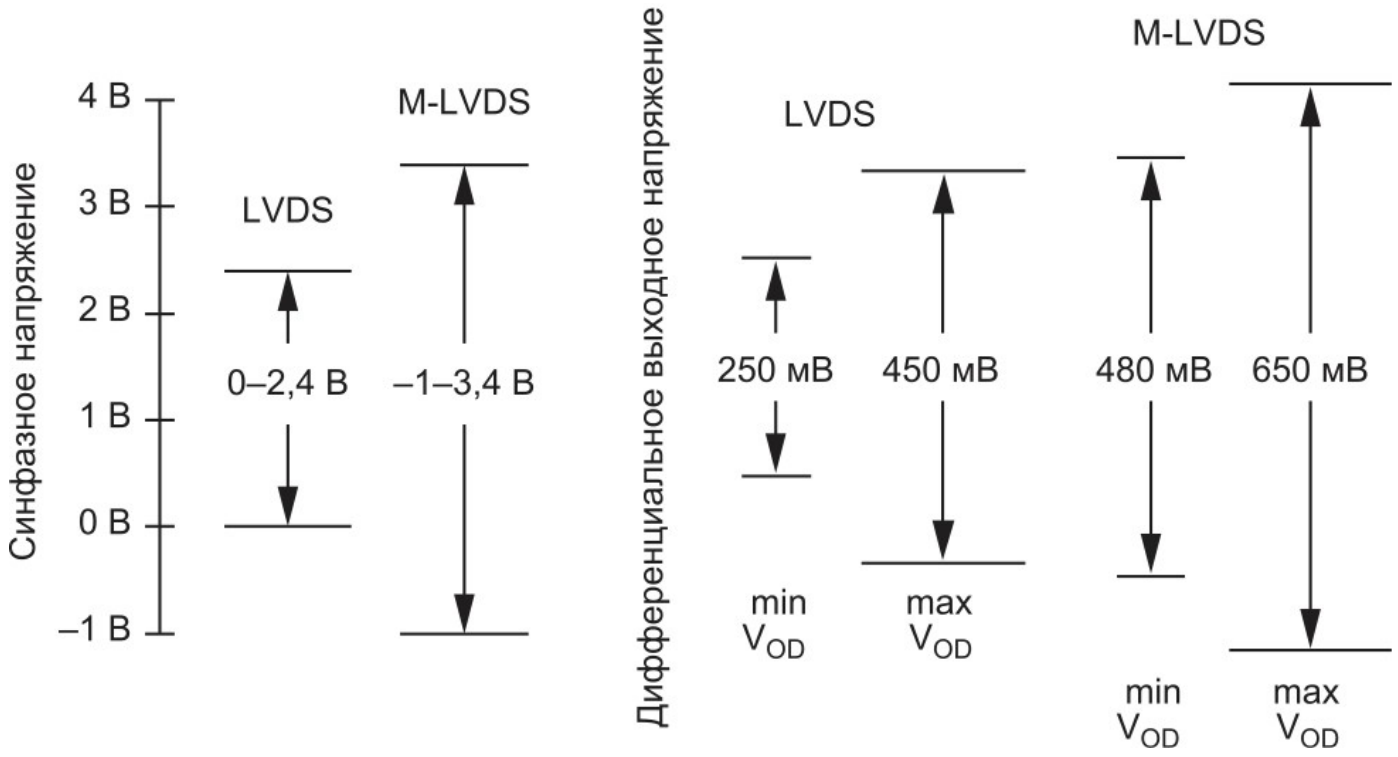


Полудуплексная шина M-LVDS

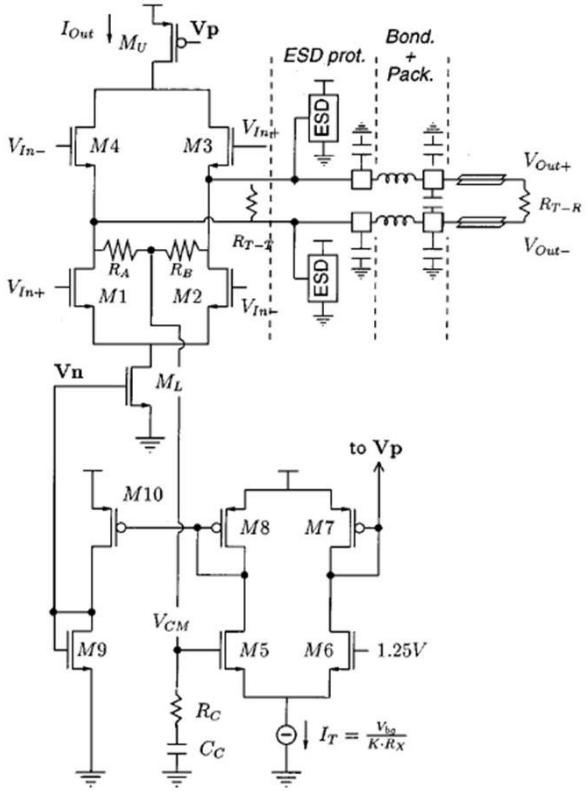
Дуплексная шина M-LVDS



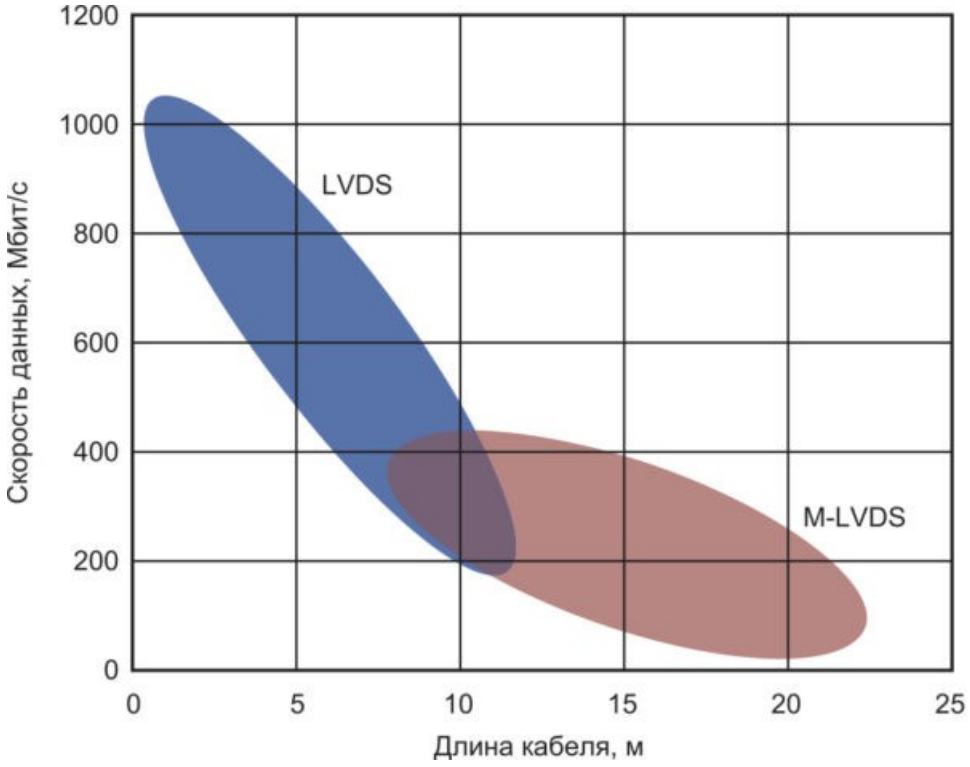
Уровни сигналов



LVDS драйвер



Скорость передачи данных



LVDS (Low-Voltage Differential Signal)

- Скорость передачи данных — до 1 Гбит/с и выше
- Пониженный уровень электромагнитных искажений
- Повышенная устойчивость к шуму
- Низкое энергопотребление
- Диапазон синфазных напряжений позволяет работать при разнице потенциалов «земли» до ± 1 В

3.2 Кодирование данных при передаче

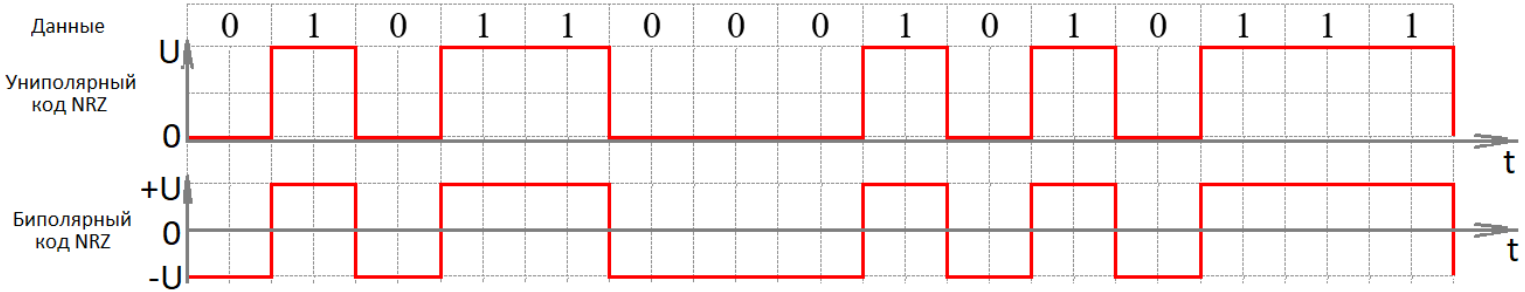
Проектирование СБИС

Физическое кодирование

- NRZ (non return to zero, без возврата к нулю)
- NRZI (non return to zero invertive, без возврата к нулю с инверсией при 1)
- RZ (return to zero, с возвратом к нулю)
- Биполярное кодирование AMI
- Manchester II (манчестерский код)
- и пр.

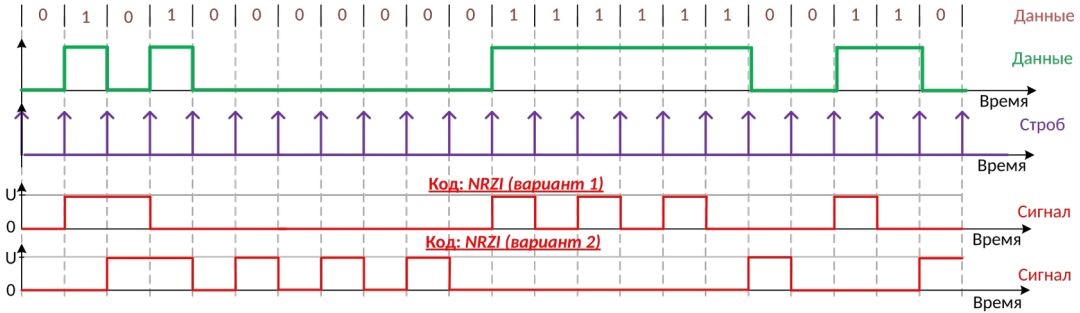
Кодирование NRZ

- Потенциальный код без возвращения к 0
- Используются два уровня потенциала:
  - 1 — положительный потенциал
  - 0 — нулевой (униполярный NRZ) или отрицательный (биполярный NRZ) потенциал
- Содержит низкочастотные компоненты (нет изменения сигнала при передачи последовательности нулей или единиц)



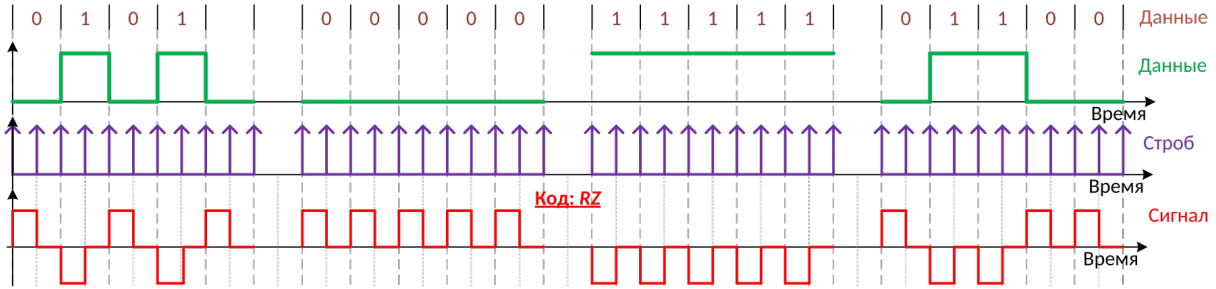
Кодирование NRZ-I

- Потенциальный код без возвращения к 0
- Используются два варианта:
  - При поступлении «1» меняется уровень потенциала, при поступлении «0» не меняется
  - При поступлении «1» не меняется уровень потенциала, при поступлении «0» меняется
- Содержит низкочастотные компоненты (нет изменения сигнала при передачи последовательности нулей или единиц)



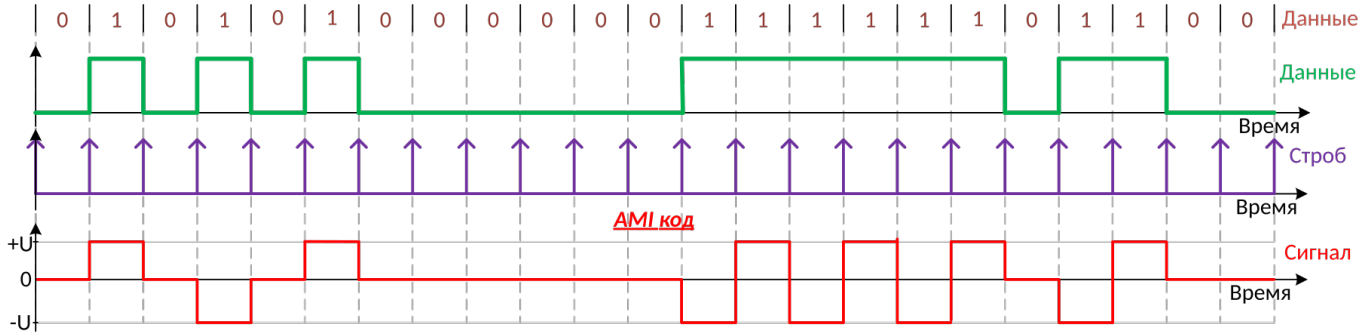
Кодирование RZ

- Потенциальный код с возвращением к 0
- Трёхуровневый код
  - При поступлении «1» меняется уровень потенциала на высокий, потом возврат в 0, при поступлении «0» меняется потенциал на низкий, потом возврат в 0
- Самосинхронизирующийся (на каждом бите информации есть возврат в 0)



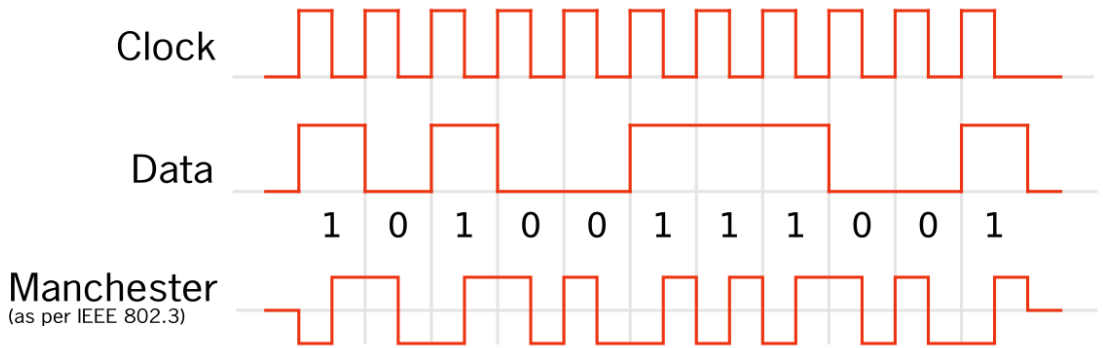
Биполярное кодирование AMI

- Alternate mark inversion
- Трёхуровневый код
  - При поступлении «1» меняется уровень потенциала на высокий, если до этого был низкий, или на низкий, если до этого был высокий
  - При поступлении «0» уровень потенциала равен 0
- Самосинхронизирующийся (однако сложность передачи последовательности нулей)



Код Manchester II

- Университет Манчестера (хранение данных на магнитном барабане компьютера Манчестерский Марк I)
- Биимпульсный код
  - Логический «0» - перепад сигнала от высокого к низкому, логическая «1» - перепад сигнала от низкого к высокому
- Самосинхронизирующийся



Области применения физического кодирования данных

Способ кодирования	Области применения
NRZ	RS-232, RS-485
NRZI	USB, FastEthernet
RZ	Ethernet, PCI Express
Биполярное AMI	Телефонные линии связи, синхронная передача по ВОЛС (заменён на PCM, pulse code modulation)
Manchester II	RFID, Ethernet

Логическое кодирование

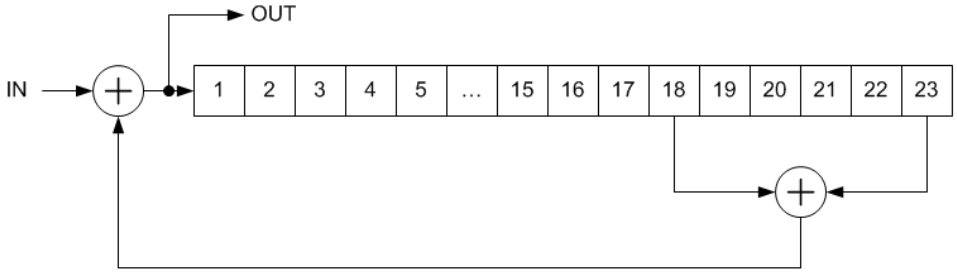
- скремблирование
- избыточные коды

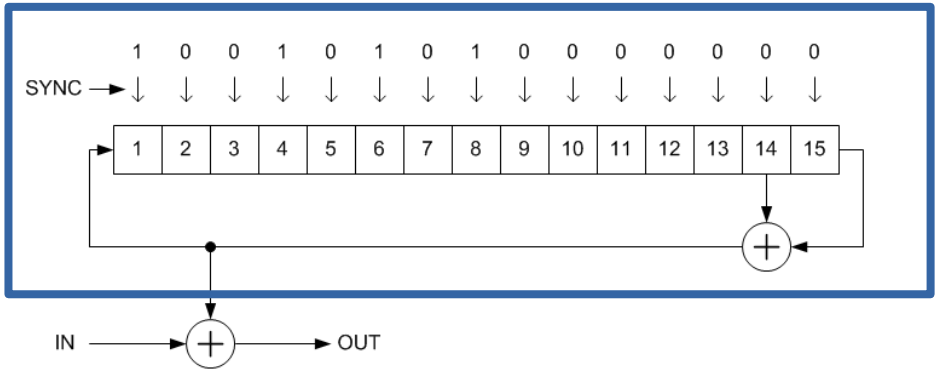
Скремблирование

- Введение синхронизирующей информации в поток данных
- Защита передаваемой информации от несанкционированного доступа

Виды скремблеров

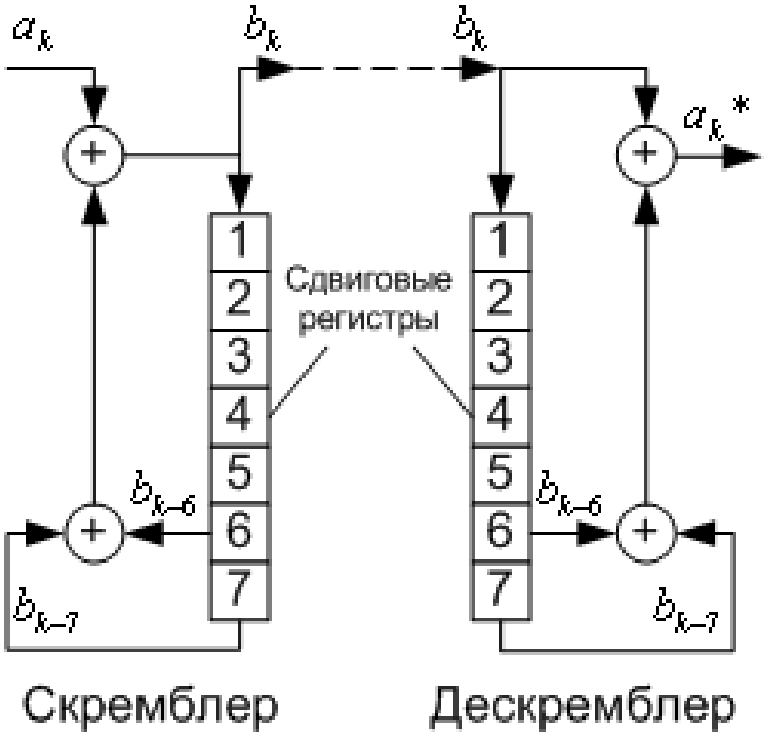
- Самосинхронизирующийся
- Аддитивный





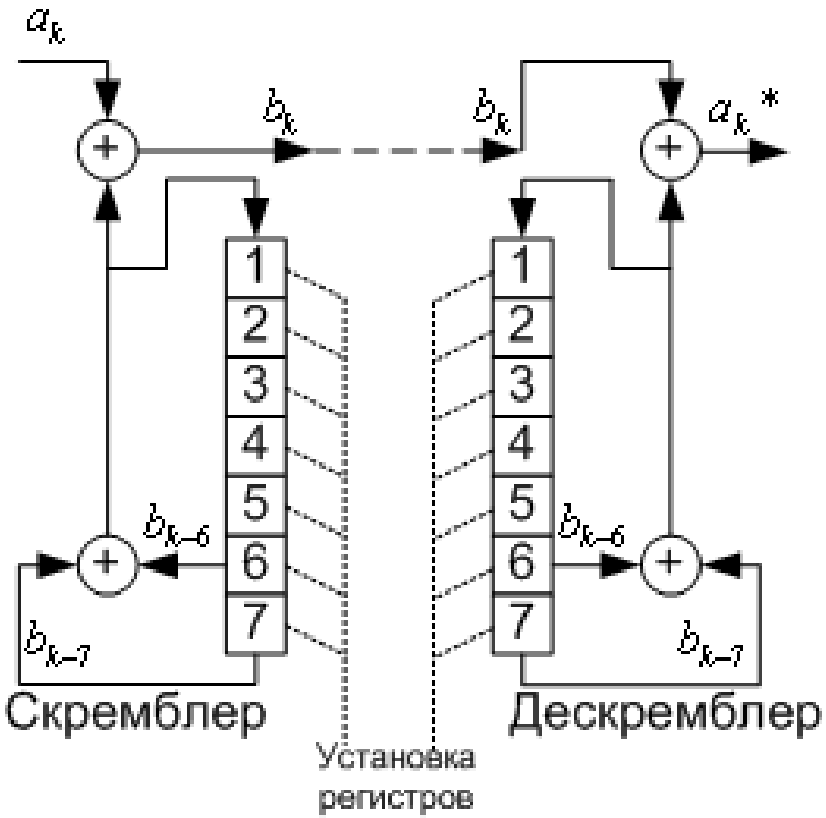
ГПСП

Самосинхронизирующийся скремблер



- «Лавинный эффект» - размножение ошибок — на 1 ошибку последовательности приходится n ошибок скремблирования (n — число обратных связей)
- Первые несколько бит не будут зашифрованы
- Нет необходимости в предустановке регистров

Аддитивный скремблер



- Нет распространения ошибок
- Требуется предустановка регистров — ключа
- Наиболее распространён

3.3 Коды с обнаружением и исправлением ошибок

Проектирование СБИС

Кодирование с защитой от ошибок

- одиночные ошибки (random error);
- групповые ошибки (purst error).

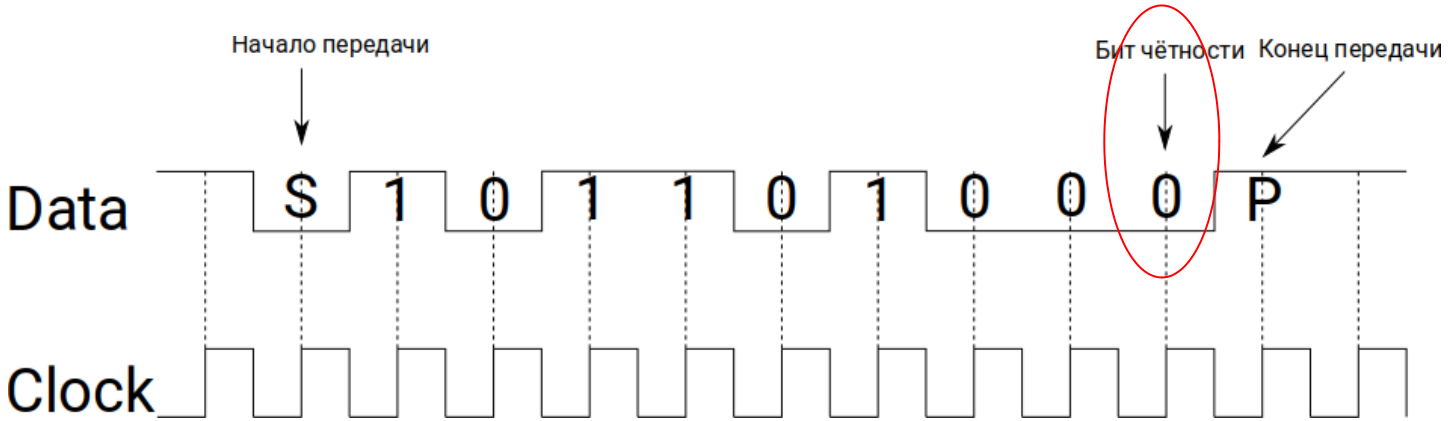
d(*x*,*y*) - расстояние Хэмминга - количество позиций в словах *x* и *y* длиной *n*, в которых *j*-е значения различны

d(1011,1101) = 2

d(1010,1011) = 1 - соседние слова

Коды с обнаружением ошибок

Применение битов чётности (нечётности) [из UART]



Особенности бита чётности

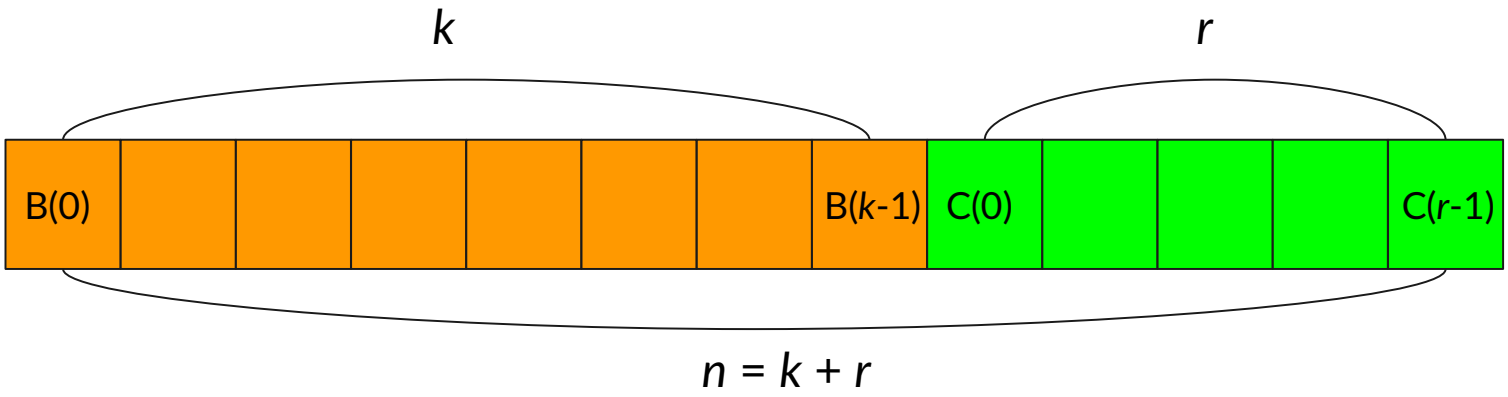
- Обнаруживает только **нечётное** количество ошибок
- Не указывает, **где** ошибка
- **Не исправляет** ошибку

Коды исправления одиночных и пакетных ошибок

forward error correction (FEC) - прямое исправление ошибок (добавление избыточных данных)

- Увеличение битов чётности
- Коды Хэмминга
- *Циклический избыточный код (CRC) - хеширование данных*
- Коды Рида-Соломона (коды БЧХ, Боуза-Чоудхури-Хоквингема)

Избыточные коды



B - передаваемая последовательность

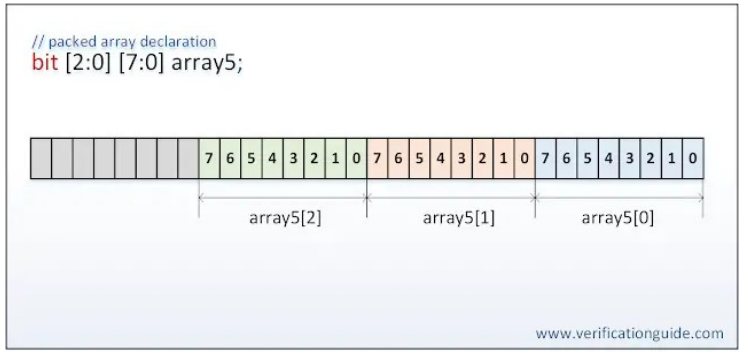
C - контрольные биты чётности

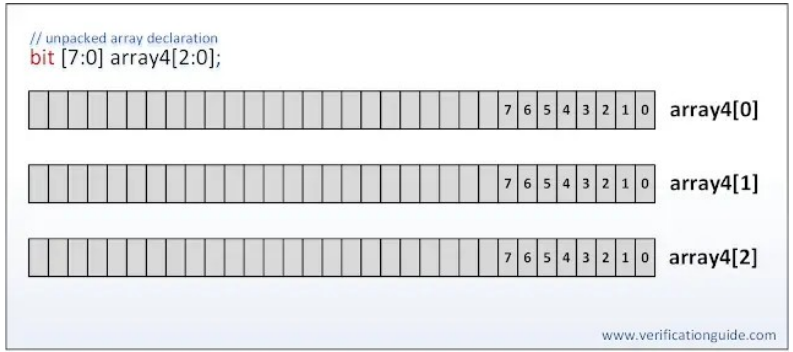
r/*n* - избыточность кода

Массивы в SystemVerilog

Массив — объединение одного типа данных

```
1  int array [10]; //10 элементов массива array типа int (32-битные)
2  bit [7:0] mem [255:0]; //256 8-битных элемента (неупакованный массив)
3  bit [255:0][7:0] mem; //упакованный массив
```





Структуры в SystemVerilog

Структура — объединение нескольких типов данных в один

```
 1  struct { //определение структуры
 2    logic [31:0] a, b;
 3    logic [ 7:0] opcode;
 4    logic [23:0] address;
 5    } Instruction_Word;
 6
 7  typedef struct { //определение структуры
 8    logic [31:0] a, b;
 9    logic [ 7:0] opcode;
10    logic [23:0] address;
11    } instruction_word_t;
12
13  instruction_word_t IW; //объявление сигнала IW как структуру
```

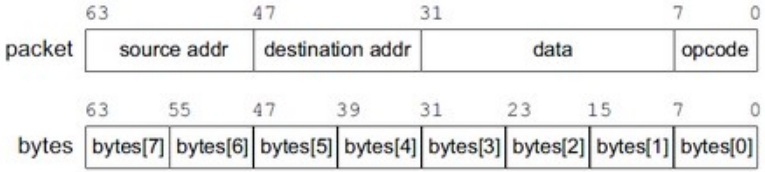
Упакованная и неупакованная структуры

```
1  struct packed {
2    logic valid;
3    logic [ 7:0] tag;
4    logic [31:0] data;
5    } data_word;
6
7  data_word.tag = 8'hf0;
8  data_word[39:32] = 8'hf0; //делаем тоже самое
```

- Неупакованная (**unpacked**) структура не эффективно использует память.
- Создаётся по умолчанию.

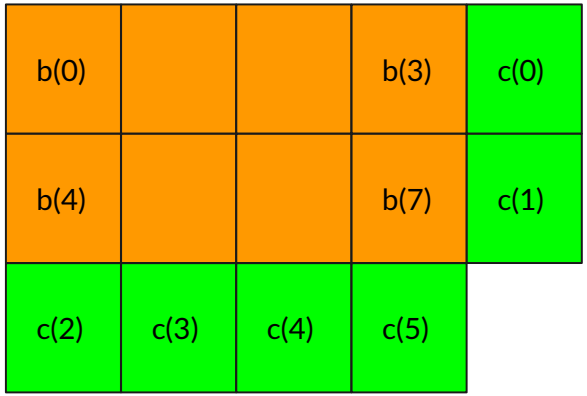
Объединения union

Объединения — это значение памяти, которое может хранить разные типы данных, но одновременно только одно. Интерес представляют **упакованные** объединения.




```
1  typedef struct packed {
2    logic [15:0] source_address;
3    logic [15:0] destination_address;
4    logic [23:0] data;
5    logic [ 7:0] opcode;
6    } data_packet_t;
7
8  union packed {
9    data_packet_t packet; // packed   structure
10   logic [7:0][7:0] bytes; // packed  array
11   } dreg;
12
13 initial begin
14   logic [15:0] src, dst;
15   for (int i = 0; i <= N; i = i + 1) begin
16     dreg.bytes[i] <= byte_in; //store as bytes
17   end
18   src = dreg.source_address;
19   dst = dreg.destination_address;
20 end
```

Вертикальная и горизонтальная чётность (блочная чётность)




```
1   module structParity(clk_i, rst_i, data_i, data_o);
2
3   input logic          clk_i;
4   input logic          rst_i;
5
6   input logic  [ 7:0] data_i;
7
8   output logic [13:0] data_o;
9
10  struct packed {
11      logic [7:0] data;
12      logic [5:0] parity;
13  } str_data;
14
15  always_ff @( posedge clk_i or negedge rst_i) begin : input_data
16      if (!rst_i) begin
17          str_data.data <= '0;
18      end else begin
19          str_data.data <= data_i;
20      end
21  end
22
23  always_comb begin : parity_calculation
24      str_data.parity = {str_data.data[7]^str_data.data[3],
25                         str_data.data[6]^str_data.data[2],
26                         str_data.data[5]^str_data.data[1],
27                         str_data.data[4]^str_data.data[0],
28                         ^str_data.data[7:4],
29                         ^str_data.data[3:0]};
30  end
31
32  always_ff @(posedge clk_i or negedge rst_i) begin : output_data
33      if (!rst_i) begin
34          data_o <= '0;
35      end else begin
36          data_o <= str_data;
37      end
38  end
39
40  endmodule
```

Особенности

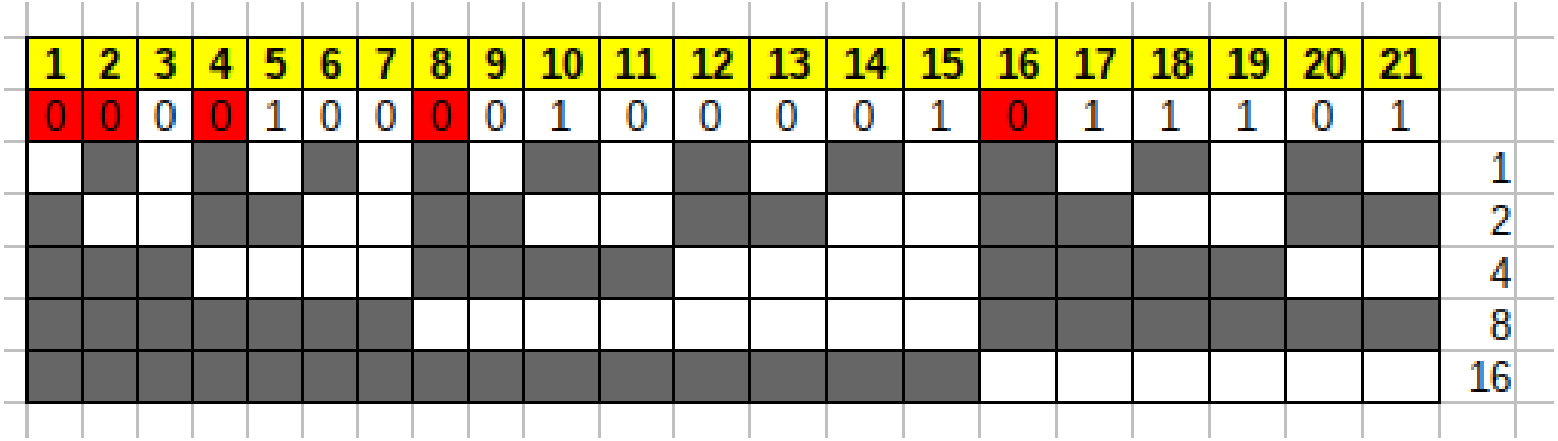
- **Простая** реализация (только XOR)
- **Обнаруживает** и **исправляет** одну ошибку
- **Большая** избыточность (на 8 бит дополнительно 6 бит)

Коды Хэмминга

- Алгоритм Хэмминга (7, 4), (15, 11), (31, 26)
- Минимальное расстояние Хэмминга = 3
- Самокорректирующийся код (SEC / DED)
  - Исправляет одну ошибку (Single Error Correction)
  - Обнаруживает две ошибки (Double Error Detection)

Принцип кодирования

Каждый бит с номером *N* является битом чётности для каждых *N* последовательных бит последовательности, стоящих через *N* последовательных бит, начиная с *N*-го бита



В общем случае бит b_N будет проверяться битами b_1, b_2, …, b_j, такими, что $1 + 2 + \ldots + j = N$. Например, бит 5 проверяется 1 и 4 битом.

Кодирование (7, 4) и синдром ошибок

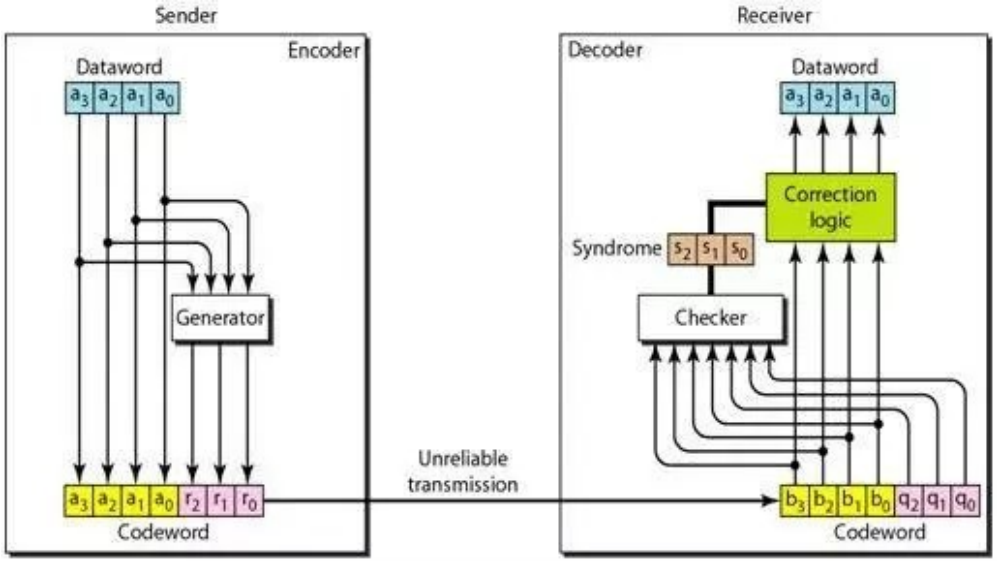
$$y_5 = x_1 \oplus x_2 \oplus x_3,$$
$$y_6 = x_2 \oplus x_3 \oplus x_4,$$
$$y_7 = x_1 \oplus x_2 \oplus x_4.$$

$$S1 = y_1 \oplus y_2 \oplus y_3 \oplus y_5$$
$$S2 = y_2 \oplus y_3 \oplus y_4 \oplus y_6$$
$$S3 = y_1 \oplus y_2 \oplus y_4 \oplus y_7$$

Синдром			Маска ошибки						
s3	s2	s1	y4/ x4	y3/ x3	y2/ x2	y1/ x1	y7/ c3	y6/ c2	y5/ c1
1	1	0	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0
1	0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1

https://zvondozvon.ru/radiosvyaz/kody-hemminga

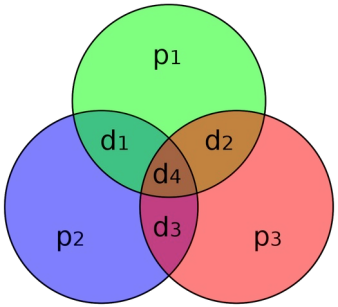
Функциональные блоки




```
1  function bit [6:0] mask (logic [2:0] syndrome);
2    case (syndrome)
3      3'b110  : mask = 7'b1000000;
4      ...
5      default : mask = 7'b0000000;
6    endcase
7  endfunction
8
9  assign data_new = data_old ^ mask(syndrome);
```

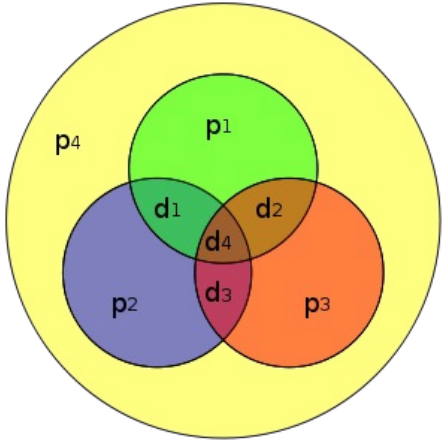
https://krazytech.com/projects/implementation-of-hamming-code

Обнаружение двух ошибок (SEC-DED)



Однократная ошибка:

- **синдром** ненулевой
- **расширенный бит чётности** изменён



Двукратная ошибка:

- **синдром** ненулевой
- **расширенный бит чётности** НЕ изменён

Особенности

- **Обнаруживает** и **исправляет** одну ошибку
- **Обнаруживает** две ошибки, но **не исправляет** вторую
- Меньшая избыточность

k	*r*
2-4	3
5-11	4
12-26	5
27-57	6

Циклический избыточный код (CRC)

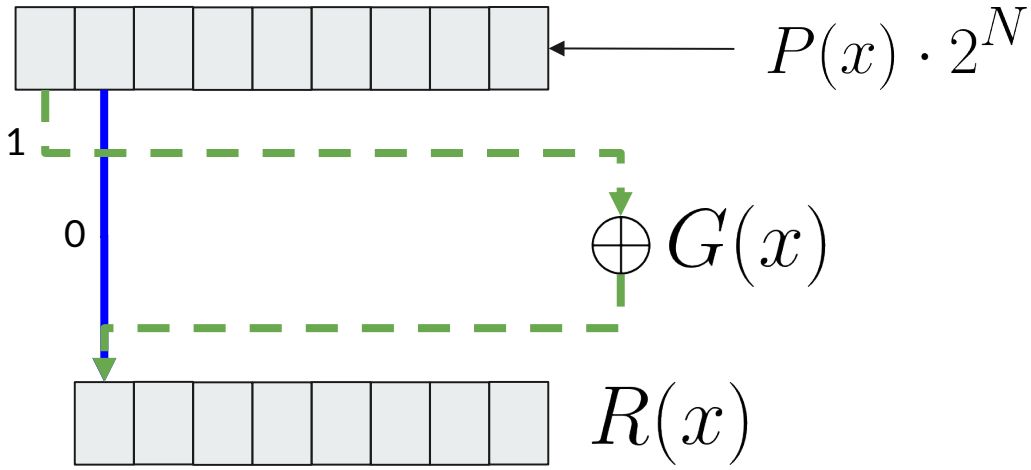
Деление передаваемого многочлена $P(x)$ на порождающий $G(x)$. Код - остаток от деления $R(x)$

$$R(x) = \left(P(x) \cdot x^N \right) \mod G(x)$$

Алгоритм

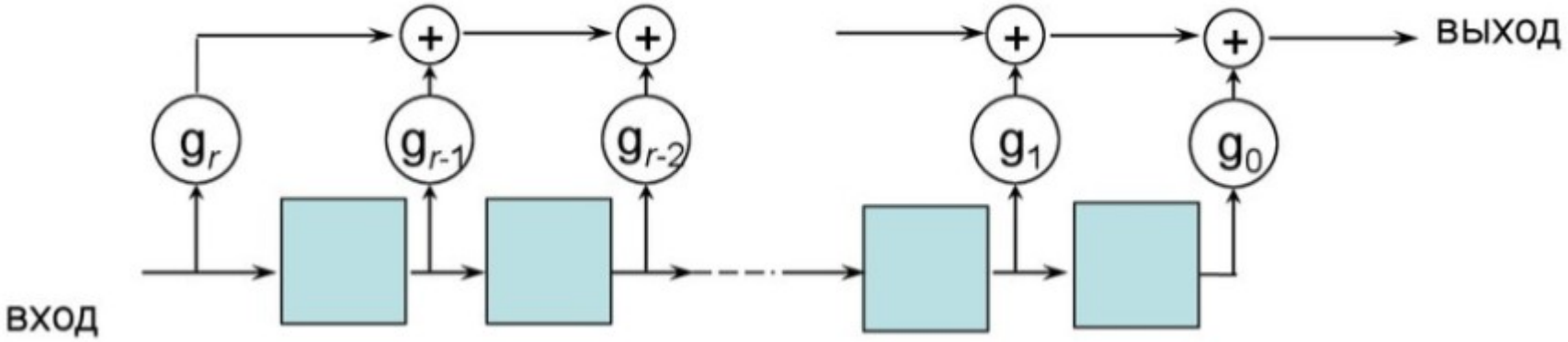
1. К передаваемой посылке дописывается N нулей
2. Слово сдвигается влево с последующей операцией XOR, если старший разряд “1”
3. Слово сдвигается влево без последующей операции XOR, если старший разряд “0”
4. Когда все разряды прошли, итоговое значение - CRC

Иллюстрация



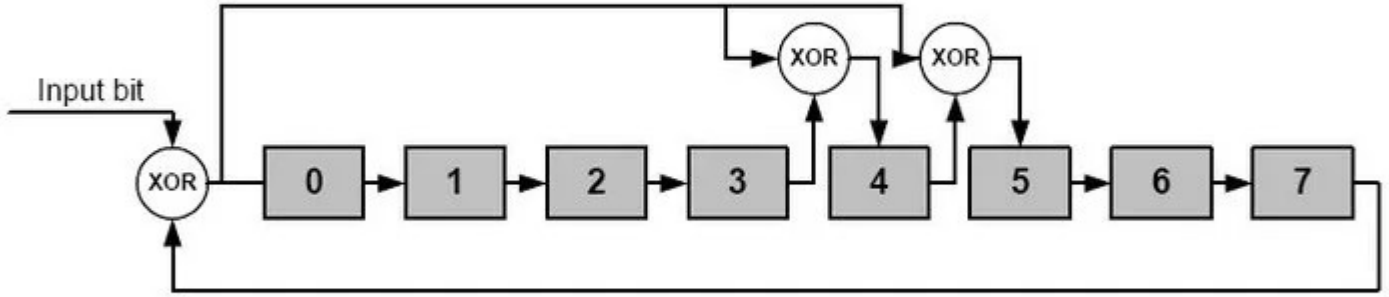
Аппаратная реализация кодирования

Умножение на порождающий многочлен (g_i = {0, 1})



Кодирование

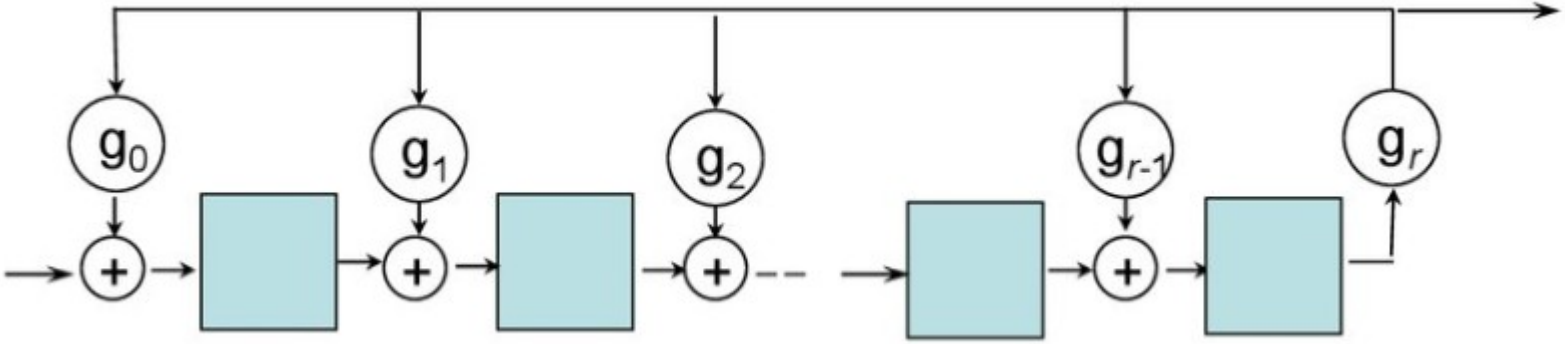
polynomial : $x^8 + x^5 + x^4 + 1$



CRC- 8 generator

Аппаратная реализация декодирования

Деление на порождающий многочлен



Особенности

- **Простота** реализации
- **Не исправляет** ошибку
- **Невысокие** затраты ресурсов
- **Сформированный** математический аппарат

Коды Рида-Соломона

https://ru.wikipedia.org/wiki/Код_Рида_—_Соломона

- Кодирование
  - умножение на **G**(*x*)
  - добавление 2*t* нулей
  - деление на **G**(*x*)
  - остаток от деления **R**(*x*) - RS-код

- Декодирование
  - деление на **G**(*x*)
  - остаток 0? нет ошибок : вычисление синдрома ошибок
  - нахождение полинома локаторов ошибок
  - корни полинома - нахождение местоположения ошибок
  - вычисление значений ошибок и построение полинома ошибок
  - исправление ошибок

Особенности

- **Обнаруживает** и **исправляет** ошибки в блоках данных
- **Высокие** затраты ресурсов

Задание на КМ-4, 5, 6

- Реализовать передатчик, который передаёт закодированную и скремблированную последовательность через сериалайзер и десериалайзер с последующим физическим кодированием.
- Реализовать приёмник, который декодирует принимаемую последовательность в вид физического кодирования без возврата к нулю (NRZ), декодированием с обнаружением ошибки и дескремблированием.
- Продемонстрировать приём и передачу последовательности без ошибки, а также с ошибкой (с запросом высылки последовательности повторно или илсправлением).
- Привести доказательство верного результата скремблирования и генерации кодовых бит (формирование golden model любым способом: вручную, при помощи языков программирования).

Вид сериалайзера / десериалайзера

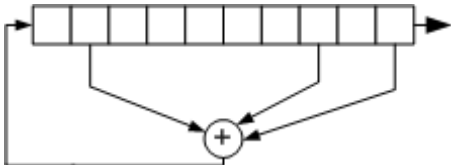
Данные Din[n-1:0] сопровождаются тактовым сигналом Cin (передний фронт). После формирования n/m-ой посылки должен формироваться импульс Done (передача завершена).
Генератором случайных чисел определить вид SerDes.

Вариант	Вид SerDes (n / m)
1	28 / 4
2	21 / 3
3	16 / 4
4	32 / 8
5	44 / 4

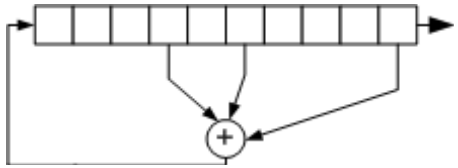
Варианты скремблеров

Генератором случайных чисел определить тип скремблера — самосинхронизирующийся (нечётное число) или аддитивный (чётное число).
Аддитивные скремблеры. Генератором случайных чисел определить вид сдвигового регистра. Для определения начального значения регистра использовать генератор случайных чисел.
Для самосинхронизирующегося скремблера произвести преобразование схемы.

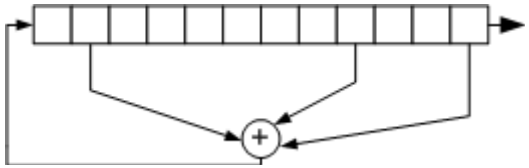
1



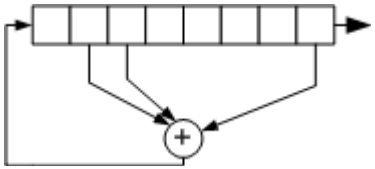
2



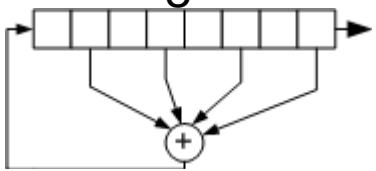
3



4



5



Варианты кодировщиков

Генератором случайных чисел определить тип кодировщика: Хэмминга (нечётное число) или CRC (чётное число).
Используя генератор случайных чисел, определить порождающий полином для CRC.

Вариант	Порождающий полином для CRC
1	11101
2	11011
3	10111
4	10101
5	10011

Физическое кодирование

Генератором случайных чисел определить вид физического кодирования.

Вариант	Вид кодирования
1	Manchester II (униполярный)
2	NRZI (с инверсией при 1)
3	NRZI (с инверсией при 0)