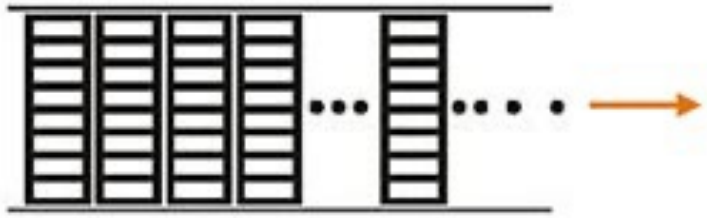
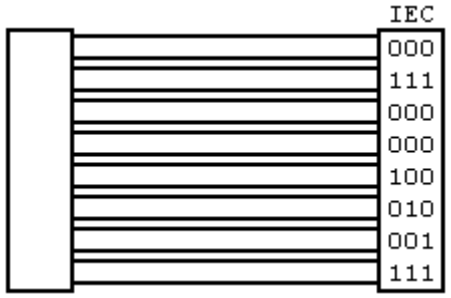
Распознавание звукового тона методом zero-crossing.

Интерфейс SPI.

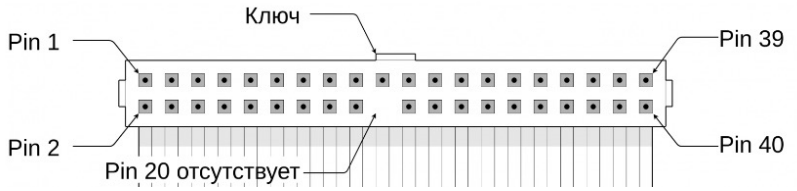
Последовательные и параллельные интерфейсы передачи данных

Параллельная передача данных





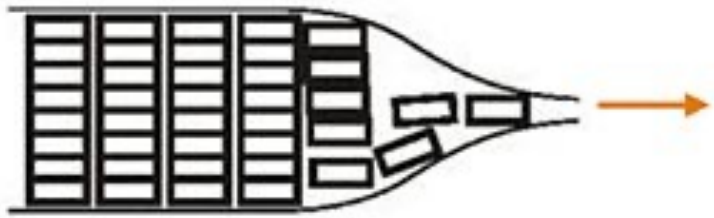
1. *Потенциально бОльшая скорость передачи.*
2. *Проблема передачи данных на большие расстояния*
3. *Искажения сообщений*



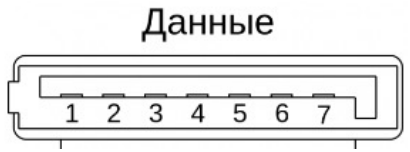
Пример: PATA

ISA ATA SCSI PCI

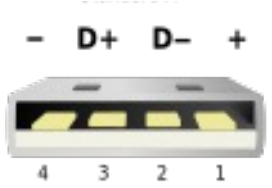
Последовательная передача данных



1. Усложнение логики приёма – передачи
2. Помехоустойчивость



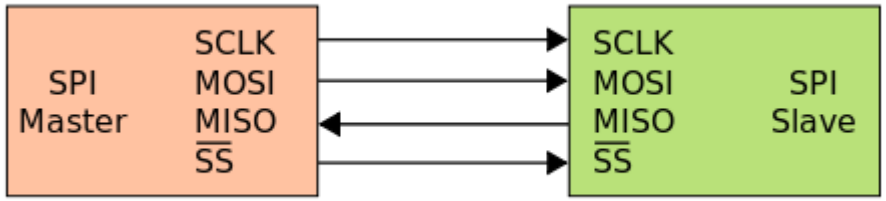
Пример: SATA

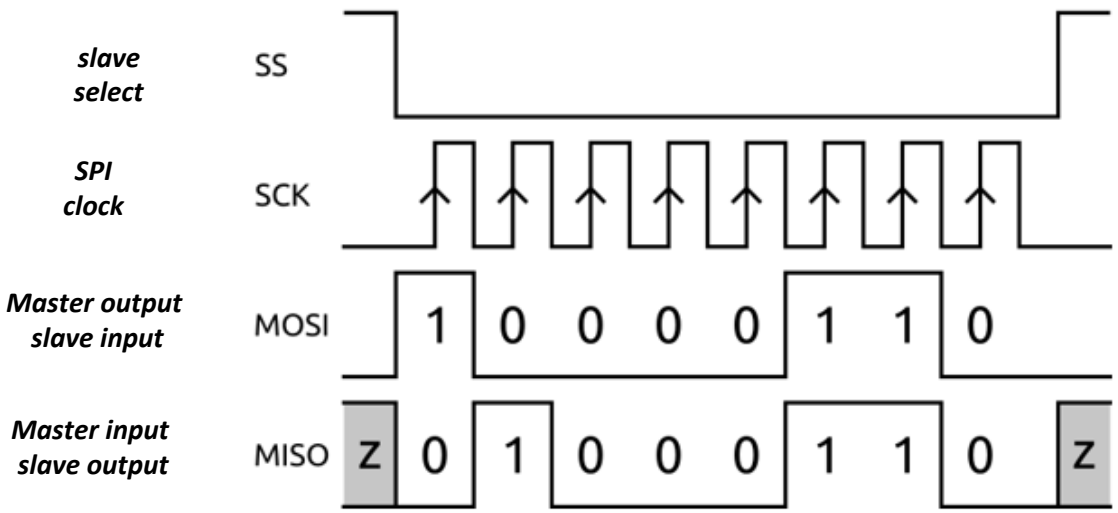


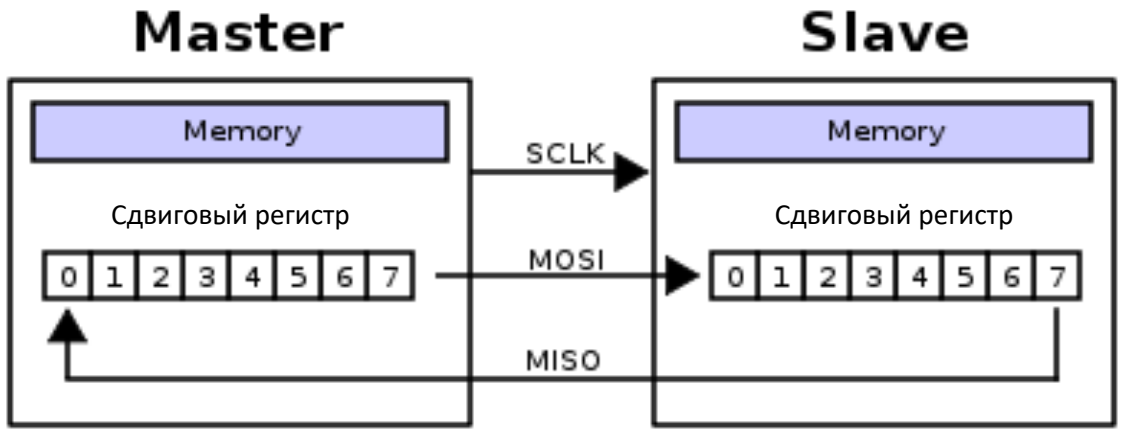
Пример: USB

RS-232 RS-422 RS-423 RS-485 I²C SPI USB PCI-E SATA

Интерфейс SPI (Serial Peripheral interface)

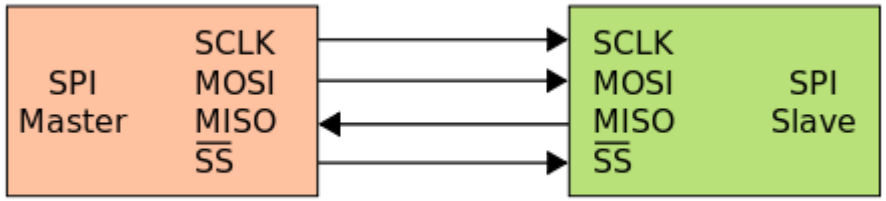






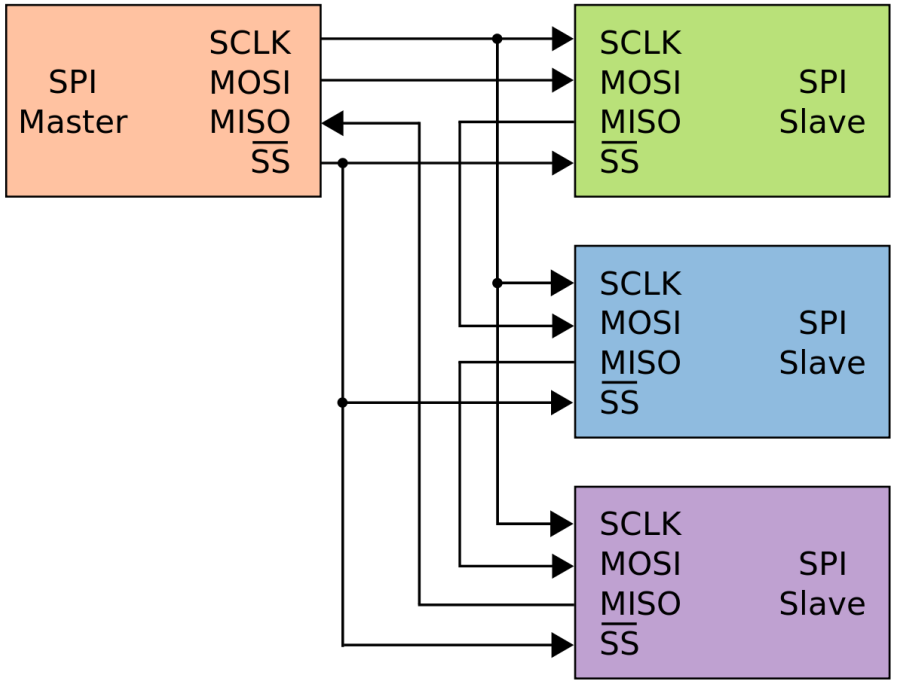
Интерфейс используется для связи с датчиками, дисплеями, микросхемами ЦАП и АЦП, RFID-ридерами, модулями беспроводной связи, включая приемо-передатчики WiFi и Bluetooth, GPRS-адаптерами и так далее.

Интерфейс SPI. Примеры подключения ведомых устройств

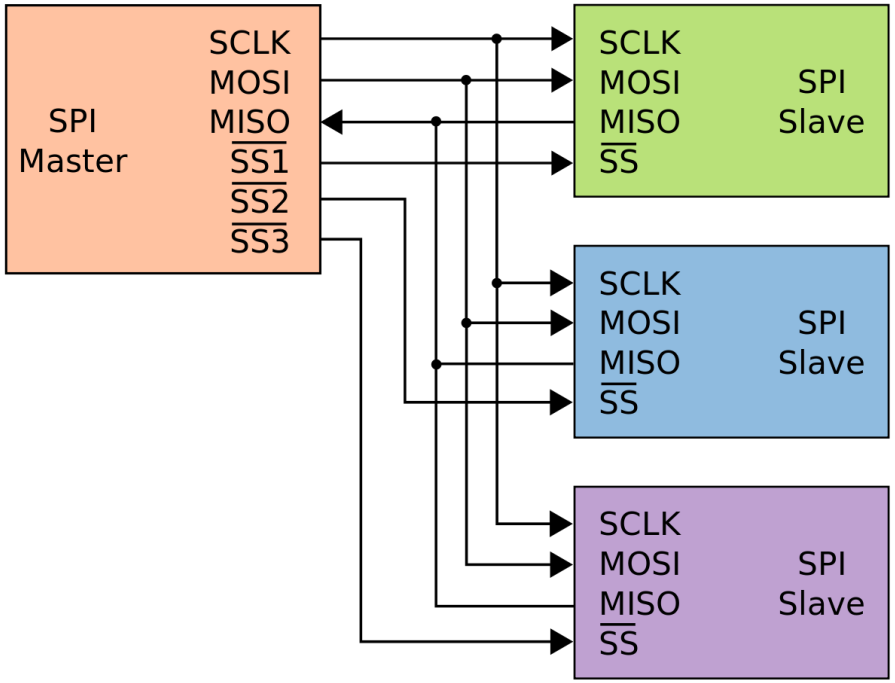


Мы можем подключить несколько устройств к основному модулю

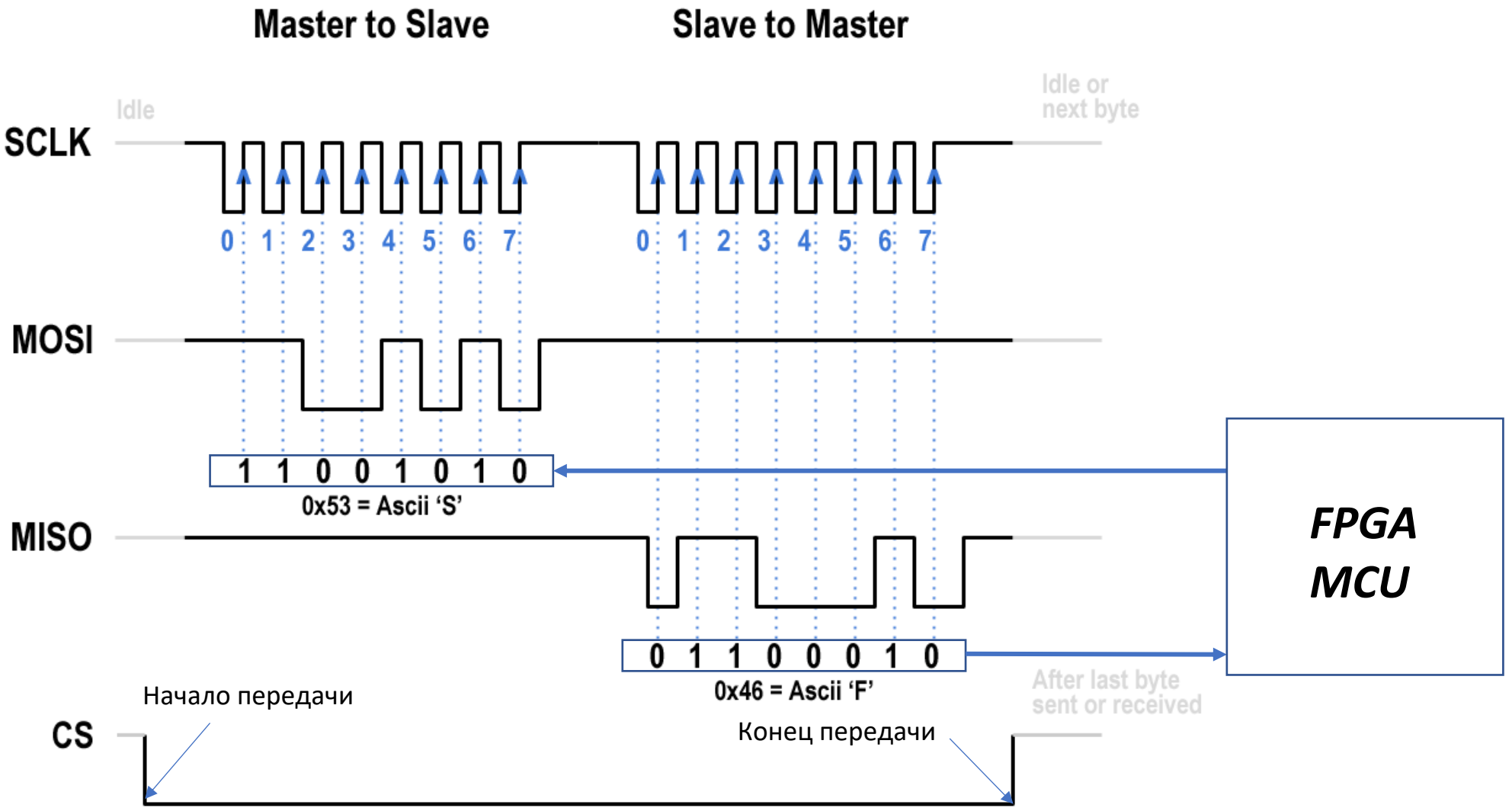
Кольцевая структура связи с несколькими ведомыми устройствами через SPI



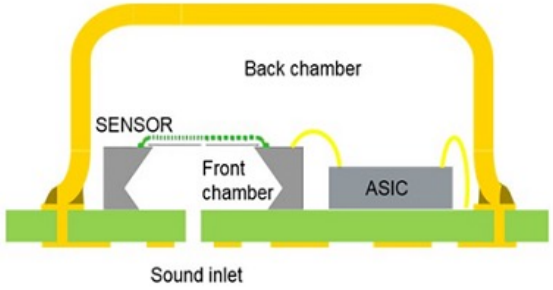
Радиальная структура связи с несколькими ведомыми устройствами через SPI

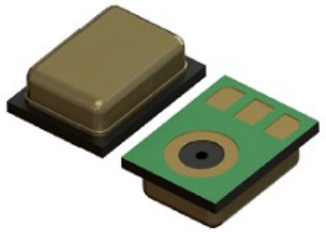


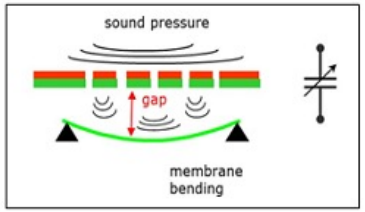
Пример передачи ASCII -символов по SPI

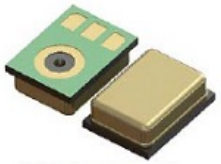


Digilent PMOD MIC3







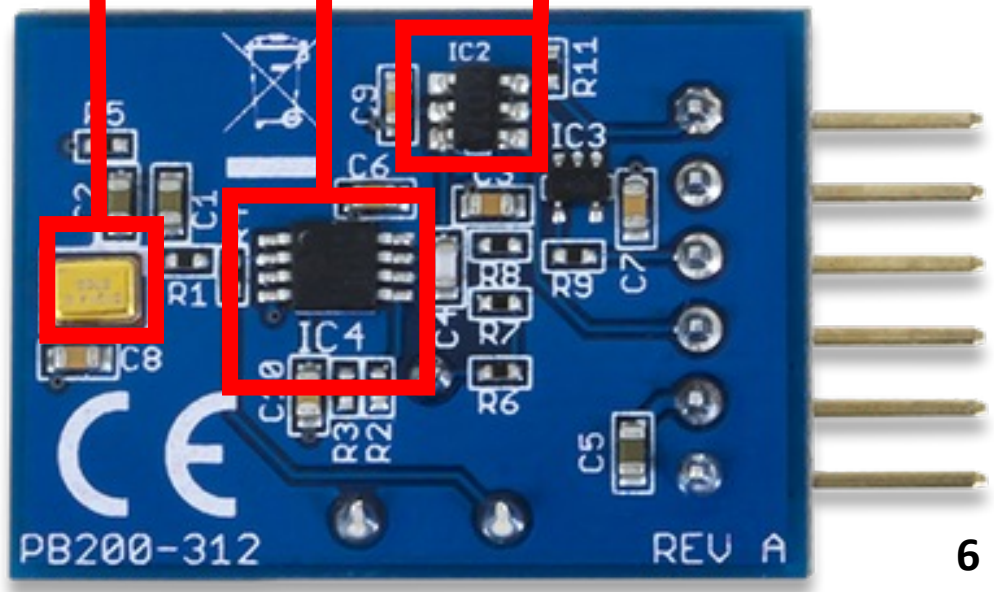


Регулировка чувствительности

Усилитель

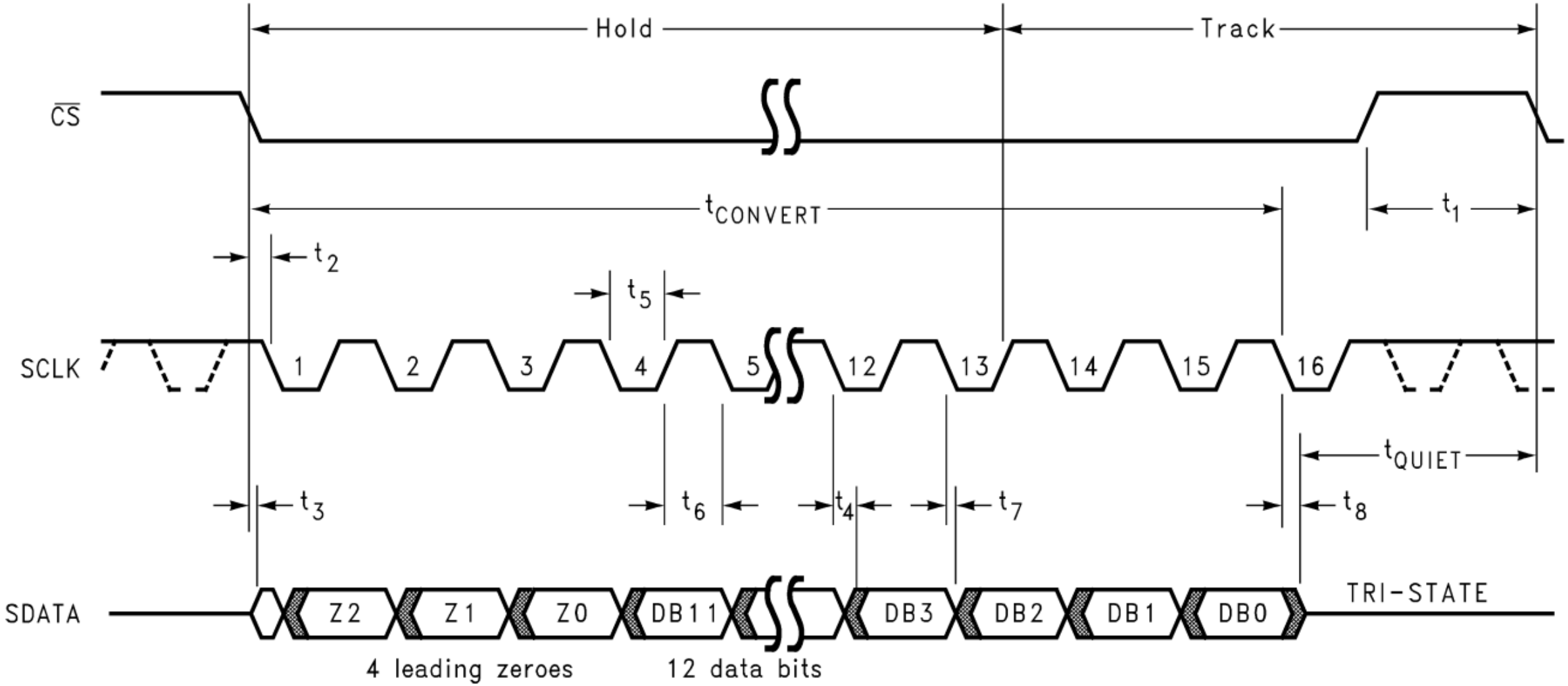
12-битный послед. АЦП

MEMS-Микрофон

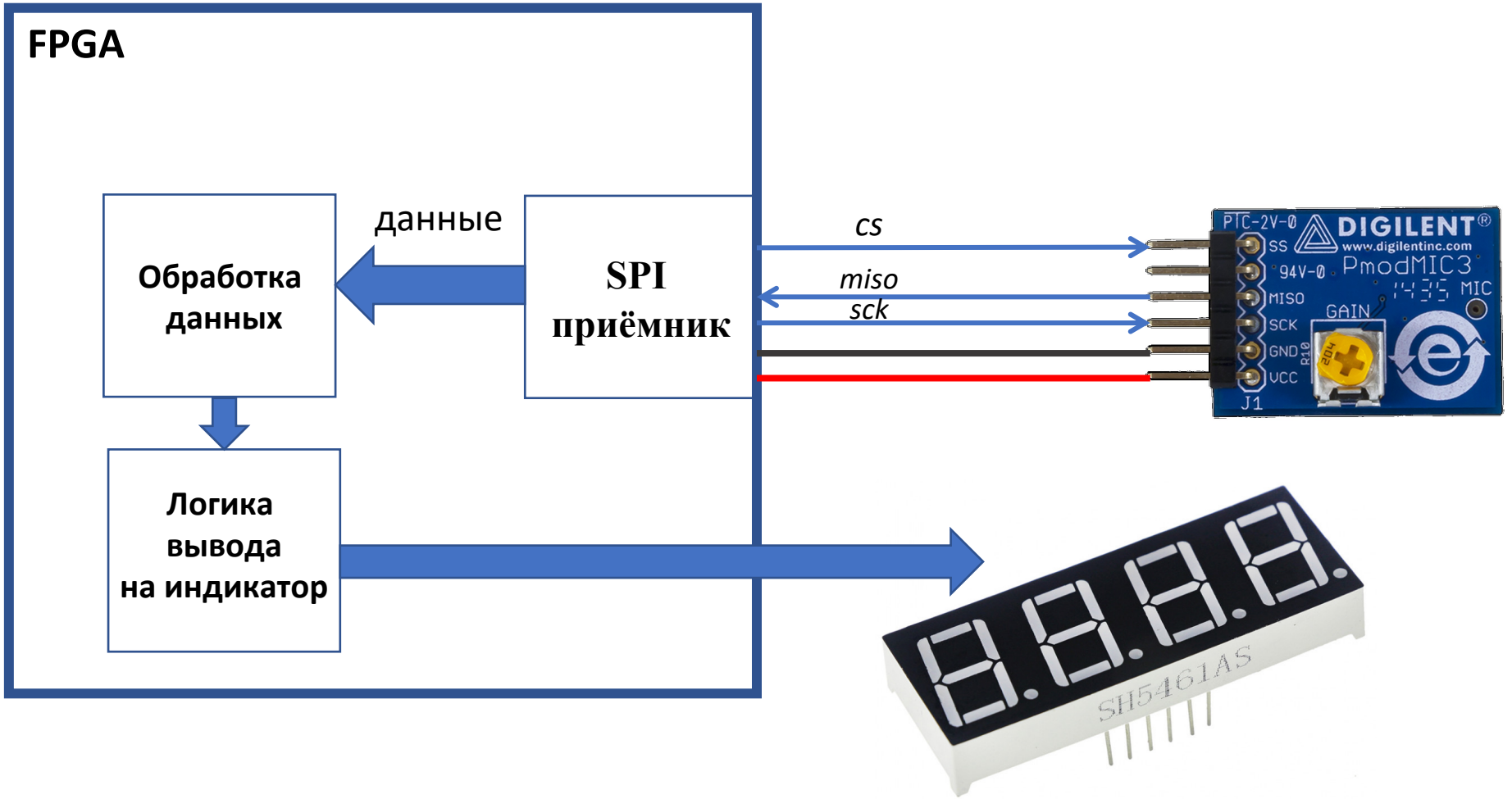


Digilent PMOD MIC3. Тактовая диаграмма АЦП

Тактовая диаграмма из datasheet на микросхему АЦП ADC7476



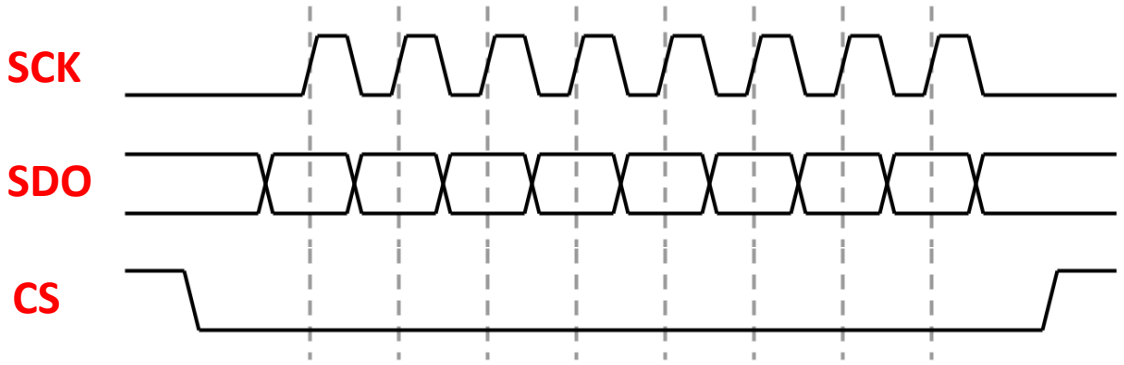
Digilent PMOD MIC3.Структура проекта

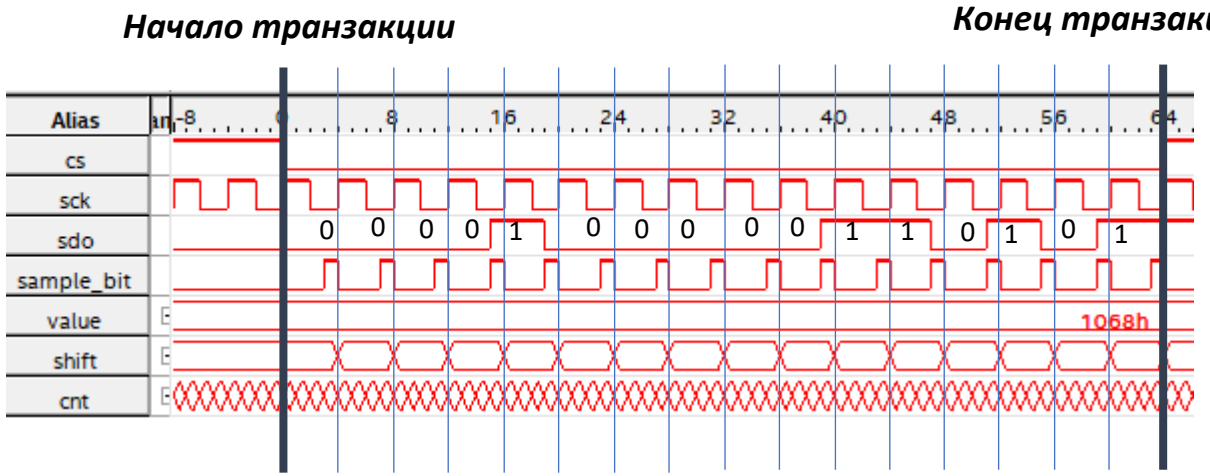


Реализация интерфейса SPI на Verilog. Формирование частот интерфейса

```
3.  module digilent_pmod_mic3_spi_receiver
4.  (
5.      input           clk,
6.      input           reset,
7.      output          cs,
8.      output          sck,
9.      input           sdo,
10.     output logic [15:0] value
11. );
```

```
14. always_ff @ (posedge clk or posedge reset)
15.     begin
16.         if (reset)
17.             cnt <= 7'b100;
18.         else
19.             cnt <= cnt + 7'b1;
20.     end
21.     assign sck = ~ cnt [1];
22.     assign cs  =   cnt [6];
```

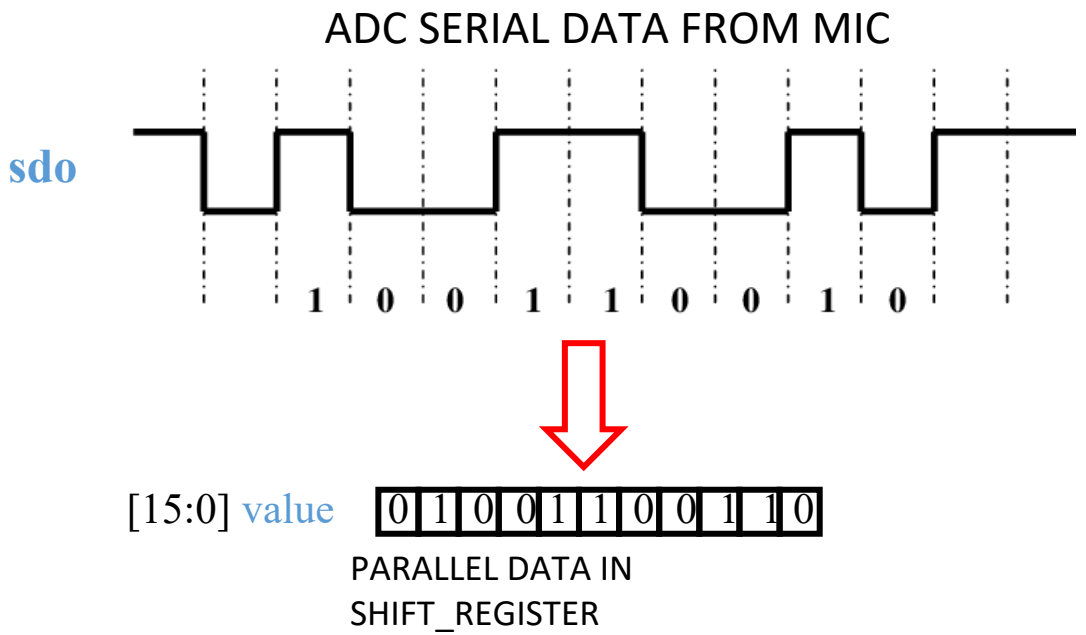


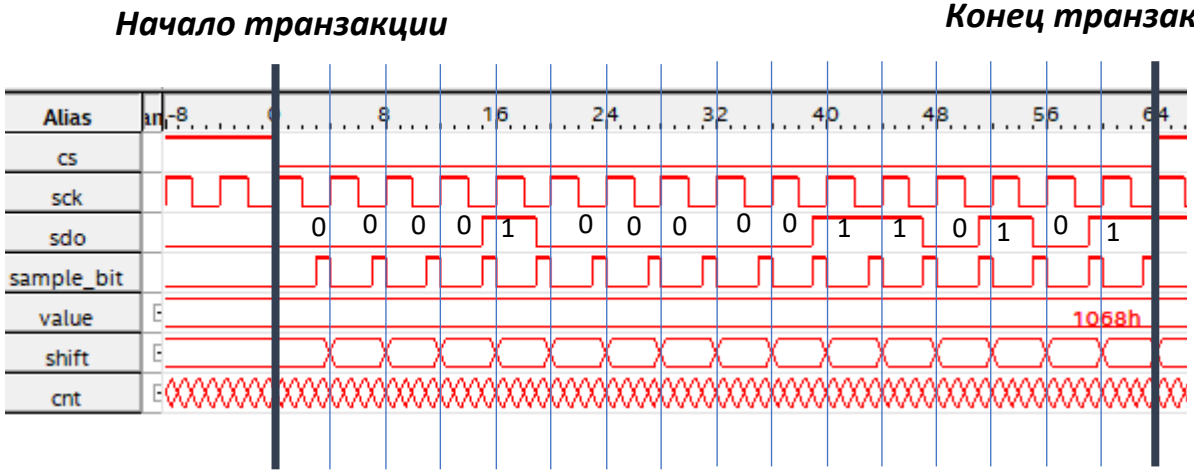


day_6/lab/pmod_mic3_spi_receiver.sv*

Реализация интерфейса SPI на Verilog. Загрузка данных

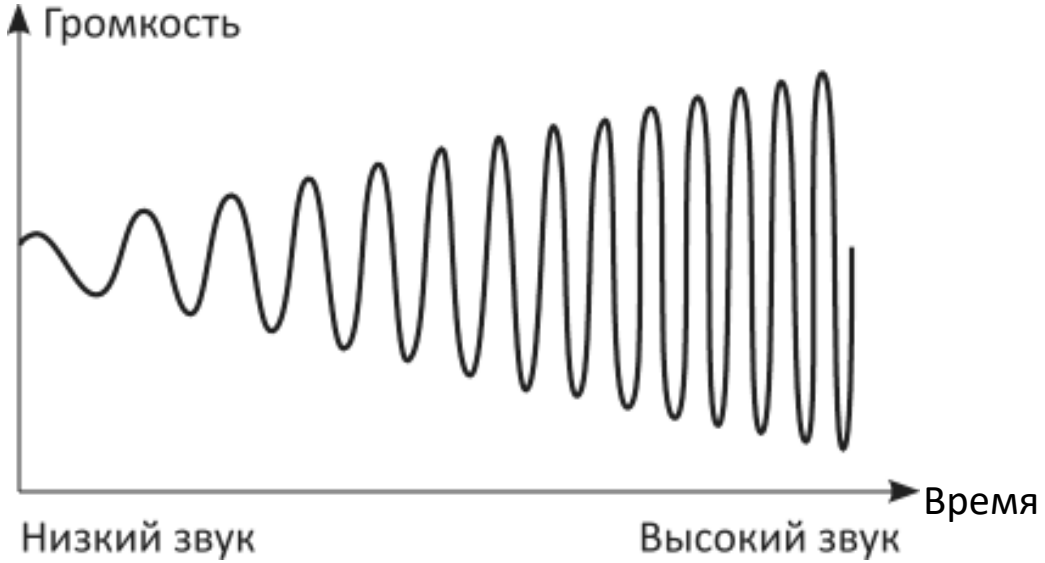
```
25.   wire sample_bit = ( cs == 1'b0 && cnt [1:0] == 2'b11 );
26.   wire value_done = ( cnt [6:0] == 7'b0 );
27.   always_ff @ (posedge clk or posedge reset)
28.   begin
29.      if (reset)
30.      begin
31.         shift <= 16'h0000;
32.         value <= 16'h0000;
33.      end
34.      else if (sample_bit)
35.      begin
36.         shift <= (shift << 1) | sdo; // shift_register
37.      end
38.      else if (value_done)
39.      begin
40.         value <= shift;
41.      end
42.   end
```



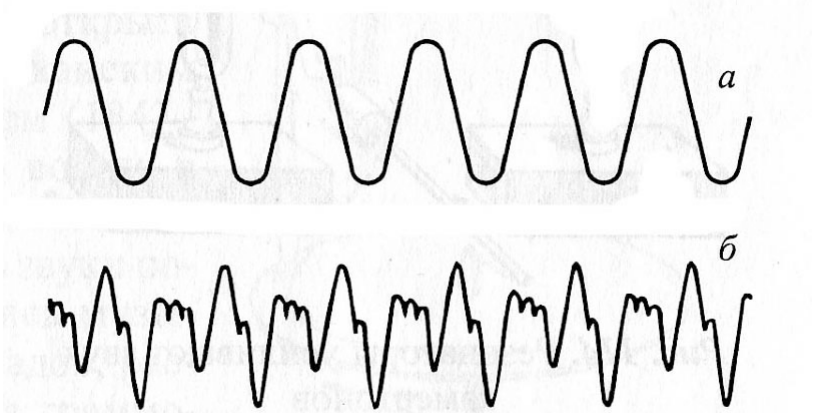


day_6/lab/pmod_mic3_spi_receiver.sv*

Немного теории музыки…



Звук – это механическое колебание определенной частоты и амплитуды



Чистый звук представляет собой, например звуковой тон

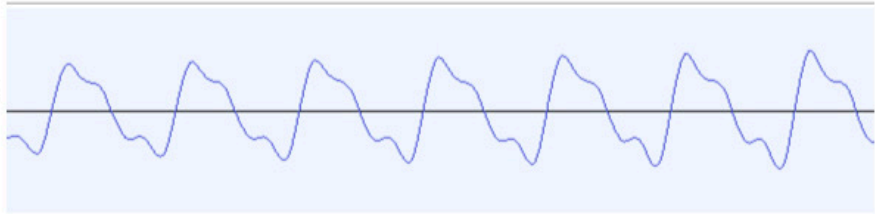
Сложный звук представляет собой сложение разных тонов, например музыкальное трезвучие

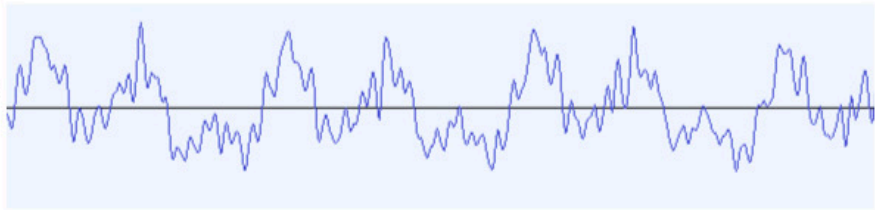
Немного теории музыки…

Каждая нота представляет собой тон колебаний определенной частоты

Ноты	Суббконтр-октава	Контр-октава	Большая	Малая	Первая	Вторая	Третья	Четвертая	Пятая
ДО	16,35	32,70	65,41	130,82	261,63	523,26	1046,52	2093,04	4186,08
ДО диез	17,32	34,65	69,30	138,59	277,18	554,36	1108,72	2217,44	4434,88
РЕ	18,35	36,71	73,42	146,83	293,66	587,32	1174,64	2349,28	4698,56
РЕ диез	19,45	38,89	77,78	155,57	311,13	622,26	1244,52	2489,04	4978,08
МИ	20,60	41,20	82,41	164,82	329,63	659,26	1318,52	2637,04	5274,08
ФА	21,83	43,65	87,31	174,62	349,23	698,46	1396,92	2793,84	5587,68
ФА диез	23,12	46,25	92,50	185,00	369,99	739,98	1479,96	2959,92	5919,84
СОЛЬ	24,50	49,00	98,00	196,00	392,00	784,00	1568,00	3136,00	6272,00
СОЛЬ диез	25,96	51,91	103,83	207,65	415,30	830,60	1661,20	3322,40	6644,80
ЛЯ	27,50	55,00	110,00	220,00	440,00	880,00	1760,00	3520,00	7040,00
ЛЯ диез	29,14	58,27	116,54	233,08	466,16	932,32	1864,64	3729,28	7458,56
СИ	30,87	61,74	123,47	246,94	493,88	987,76	1975,52	3951,04	7902,08

Нота “До”



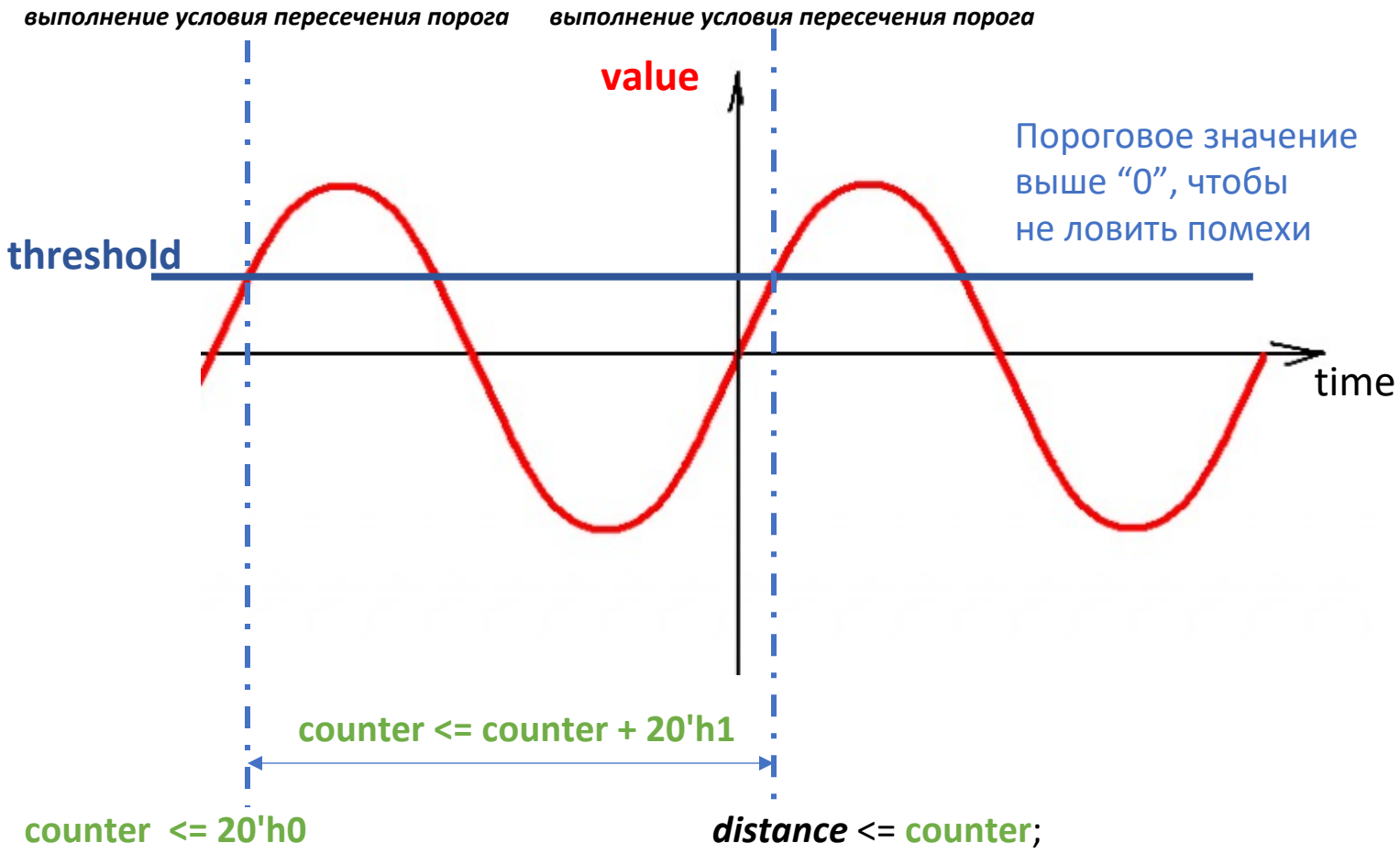


Принятое в мире обозначение нот

	до	*ре*	*ми*	*фа*	*соль*	*ля*	*си*
	C	D	E	F	G	A	B
диезы	C♯	D♯	E♯	F♯	G♯	A♯	B♯
бемоли	C♭	D♭	E♭	F♭	G♭	A♭	B♭

Обработка данных. Распознавание звукового тона методом “zero-crossing”

```
58.  localparam [15:0] threshold = 16'h1100;
59.    always_ff @ (posedge clk or posedge reset)
60.       if (reset)
61.       begin
62.          prev_value <= 16'h0;
63.          counter    <= 20'h0;
64.          distance   <= 20'h0;
65.       end
66.       else
67.       begin
68.          prev_value <= value;
69.          if (  value      >= threshold
70.             & prev_value < threshold)
71.          begin
72.            distance <= counter;
73.            counter  <= 20'h0;
74.          end
75.          else if (counter != ~ 20'h0)
76.          begin
77.            counter <= counter + 20'h1;
78.          end
79.       end
```



Отсчитываем число тактов в **counter** после пересечения **treshhold**, до следующего пересечения threshold и записываем это число в **distance.**

day_6/lab/top.sv*

Интерпретация полученных данных

```
function [19:0] high_distance (input [18:0] freq_100);
   high_distance = clk_mhz * 1000 * 1000 / freq_100 * 104;
endfunction

//----------------------------------------------------------------------

function [19:0] low_distance (input [18:0] freq_100);
  low_distance = clk_mhz * 1000 * 1000 / freq_100 * 96;
endfunction

//----------------------------------------------------------------------

function [19:0] check_freq_single_range (input [18:0] freq_100);
  check_freq_single_range =    distance > low_distance  (freq_100)
                             & distance < high_distance (freq_100);
endfunction

//----------------------------------------------------------------------

function [19:0] check_freq (input [18:0] freq_100);
  check_freq =   check_freq_single_range (freq_100 * 4)
               | check_freq_single_range (freq_100 * 2)
               | check_freq_single_range (freq_100);

endfunction
```

1. Пересчёт частоты ноты в число тактов основной частоты, равной 50 МГц по формуле:

$$distance = \frac{50\text{МГц}}{f\_note}$$

$$highdistance = \frac{50\text{МГц}}{f\_note} * \frac{104}{100}$$

$$lowdistance = \frac{50\text{МГц}}{f\_note} * \frac{96}{100}$$

2. Проверяем входит ли посчитанное число тактов в диапазон между high и low distance

3. Так же фиксируем ноты следующих двух октав, так как их частота отличается ровно в два раза

day_6/lab/top.sv (101-130)*

Интерпретация полученных данных

```
//----------------------------------------------------------------------

  wire check_C  = check_freq (freq_100_C );
  wire check_Cs = check_freq (freq_100_Cs);
  wire check_D  = check_freq (freq_100_D );
  wire check_Ds = check_freq (freq_100_Ds);
  wire check_E  = check_freq (freq_100_E );
  wire check_F  = check_freq (freq_100_F );
  wire check_Fs = check_freq (freq_100_Fs);
  wire check_G  = check_freq (freq_100_G );
  wire check_Gs = check_freq (freq_100_Gs);
  wire check_A  = check_freq (freq_100_A );
  wire check_As = check_freq (freq_100_As);
  wire check_B  = check_freq (freq_100_B );

  //----------------------------------------------------------------------

  localparam w_note = 12;

  wire [w_note - 1:0] note = { check_C  , check_Cs , check_D  , check_Ds ,
                               check_E  , check_F  , check_Fs , check_G  ,
                               check_Gs , check_A  , check_As , check_B  };
```

```
localparam [w_note - 1:0] no_note = 12'b0,

                          C  = 12'b1000_0000_0000,
                          Cs = 12'b0100_0000_0000,
                          D  = 12'b0010_0000_0000,
                          Ds = 12'b0001_0000_0000,
                          E  = 12'b0000_1000_0000,
                          F  = 12'b0000_0100_0000,
                          Fs = 12'b0000_0010_0000,
                          G  = 12'b0000_0001_0000,
                          Gs = 12'b0000_0000_1000,
                          A  = 12'b0000_0000_0100,
                          As = 12'b0000_0000_0010,
                          B  = 12'b0000_0000_0001;

localparam [w_note - 1:0] Df = Cs, Ef = Ds, Gf = Fs, Af = Gs, Bf = As;
```

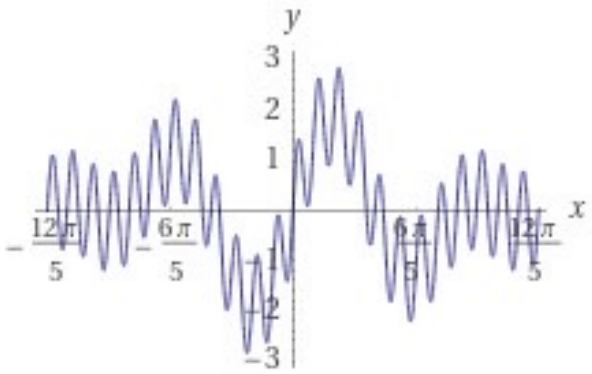
Если нота регистрируется, формируем шину из проверок, задаём параметры для каждой ноты

day_6/lab/top.sv (134-170)*

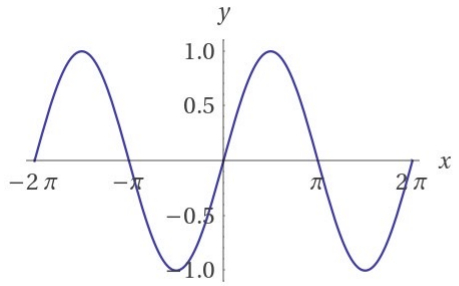
Дополнительная фильтрация

```
logic  [w_note - 1:0] d_note;  // Delayed note
  always_ff @ (posedge clk or posedge reset)
    if (reset)
      d_note <= no_note;
    else
      d_note <= note;

  logic  [17:0] t_cnt;                // Threshold counter
  logic  [w_note - 1:0] t_note;  // Thresholded note

  always_ff @ (posedge clk or posedge reset)
    if (reset)
      t_cnt <= 0;
    else
      if (note == d_note)
        t_cnt <= t_cnt + 1;
      else
        t_cnt <= 0;

  always_ff @ (posedge clk or posedge reset)
    if (reset)
      t_note <= no_note;
    else
      if (& t_cnt)
        t_note <= d_note;
```

Если нота регистрируется, проверяем, что нота удерживается достаточное число тактов основной частоты 50 МГц, за счёт чего убеждаемся что это не помеха.



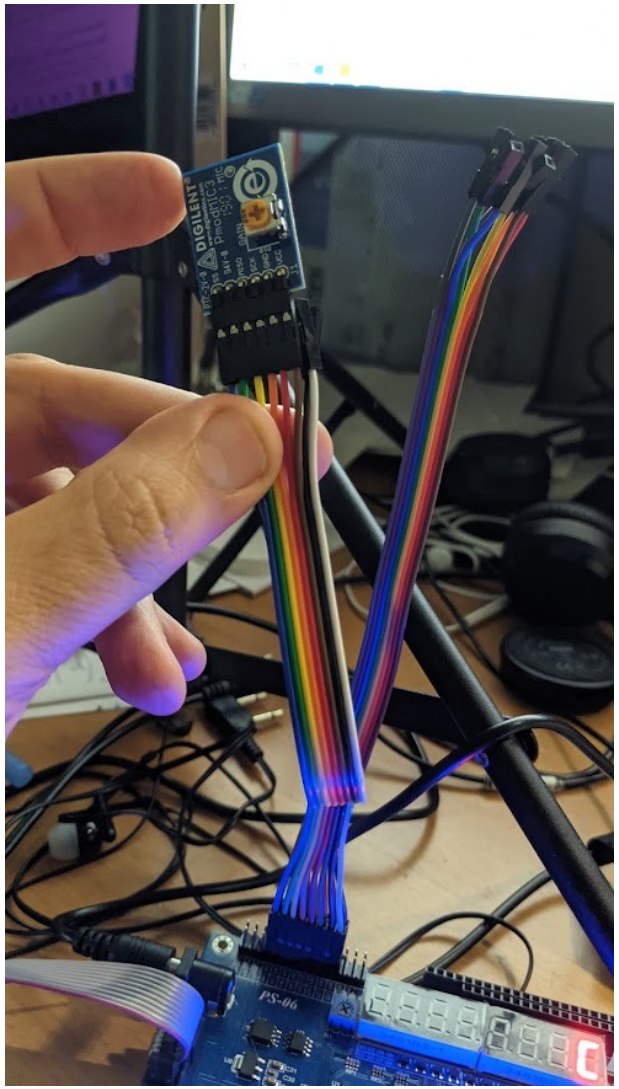
Тон с помехой



Чистый тон

day_6/lab/top.sv (134-170)*

Вывод ноты на семисегментный индикатор



```
438     always @ (posedge clk or posedge reset)
439         if (reset)
440         begin
441             abcdefgh <= 8'b11111111;
442         end
443         else if (digit_enable)
444         begin
445             if (i_digit == 3'd3)
446                 case (t_note)
447                 C  : abcdefgh <= 8'b01100011;  // C   // abcdefgh
448                 Cs : abcdefgh <= 8'b01100010;  // C#
449                 D  : abcdefgh <= 8'b10000101;  // D   //   --a--
450                 Ds : abcdefgh <= 8'b10000100;  // D#  //  |     |
451                 E  : abcdefgh <= 8'b01100001;  // E   //  f     b
452                 F  : abcdefgh <= 8'b01110001;  // F   //  |     |
453                 Fs : abcdefgh <= 8'b01110000;  // F#  //   --g--
454                 G  : abcdefgh <= 8'b01000011;  // G   //  |     |
455                 Gs : abcdefgh <= 8'b01000010;  // G#  //  e     c
456                 A  : abcdefgh <= 8'b00010001;  // A   //  |     |
457                 As : abcdefgh <= 8'b00010000;  // A#  //   --d--  h
458                 B  : abcdefgh <= 8'b11000001;  // B
459                 default : abcdefgh <= 8'b11111111;
460                 endcase
```

day_6/lab/top.sv (211-231)*

.mp3

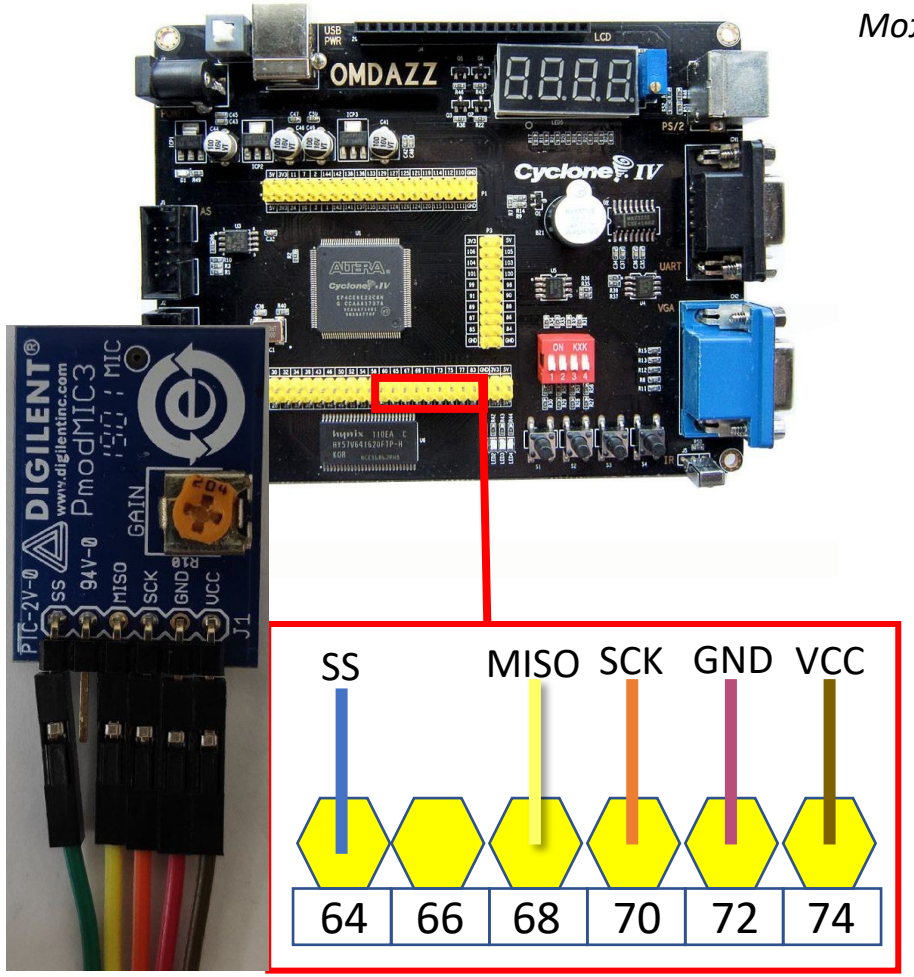
Подготовка к лабораторной работе:

1. Скачиваем приложение генератор тона на телефон
Так же можно воспользоваться заготовленными последовательностями нот в .mp3.

Можно проиграть с телефона

2. Подключаем микрофон к плате в соответствии со схемой:

3. Открываем проект в Quartus.



Лабораторная работа №1

.mp3

Можно проиграть с телефона

- **Задание 1. Заполнить таблицу параметров в соответствии с частотами нот четвертой октавы в Гц.**

Сейчас здесь правильная нота только С(До)=261.63 Гц и А(Ля)=440.00 Гц

```
87.  localparam    freq_100_C  = 26163,
88.                freq_100_Cs = 99999,
89.                freq_100_D  = 99999,
90.                freq_100_Ds = 99999,
91.                freq_100_E  = 99999,
92.                freq_100_F  = 99999,
93.                freq_100_Fs = 99999,
94.                freq_100_G  = 99999,
95.                freq_100_Gs = 99999,
96.                freq_100_A  = 44000,
97.                freq_100_As = 99999,
98.                freq_100_B  = 99999;
```

$$freq_{100} = f_{note} * 100$$

$$distance = \frac{50\text{МГц}}{f\_note}$$

Ноты	Суббконтр-октава	Контр-октава	Большая	Малая	Первая (f_{note})
ДО	16,35	32,70	65,41	130,82	261,63
ДО диез	17,32	34,65	69,30	138,59	277,18
РЕ	18,35	36,71	73,42	146,83	293,66
РЕ диез	19,45	38,89	77,78	155,57	311,13
МИ	20,60	41,20	82,41	164,82	329,63
ФА	21,83	43,65	87,31	174,62	349,23
ФА диез	23,12	46,25	92,50	185,00	369,99
СОЛЬ	24,50	49,00	98,00	196,00	392,00
СОЛЬ диез	25,96	51,91	103,83	207,65	415,30
ЛЯ	27,50	55,00	110,00	220,00	440,00
ЛЯ диез	29,14	58,27	116,54	233,08	466,16
СИ	30,87	61,74	123,47	246,94	493,88

До	До#	Ре	Ре#	Ми	Фа	Соль	Соль#	Ля	Ля#	Си	До
C	C_s	D	D_s	E	F	G	G_s	A	A_s	B	C

- **Задание 2*. Вывести на семисегментные индикаторы буквы С О Л Ь, когда микрофон распознает ноту G**

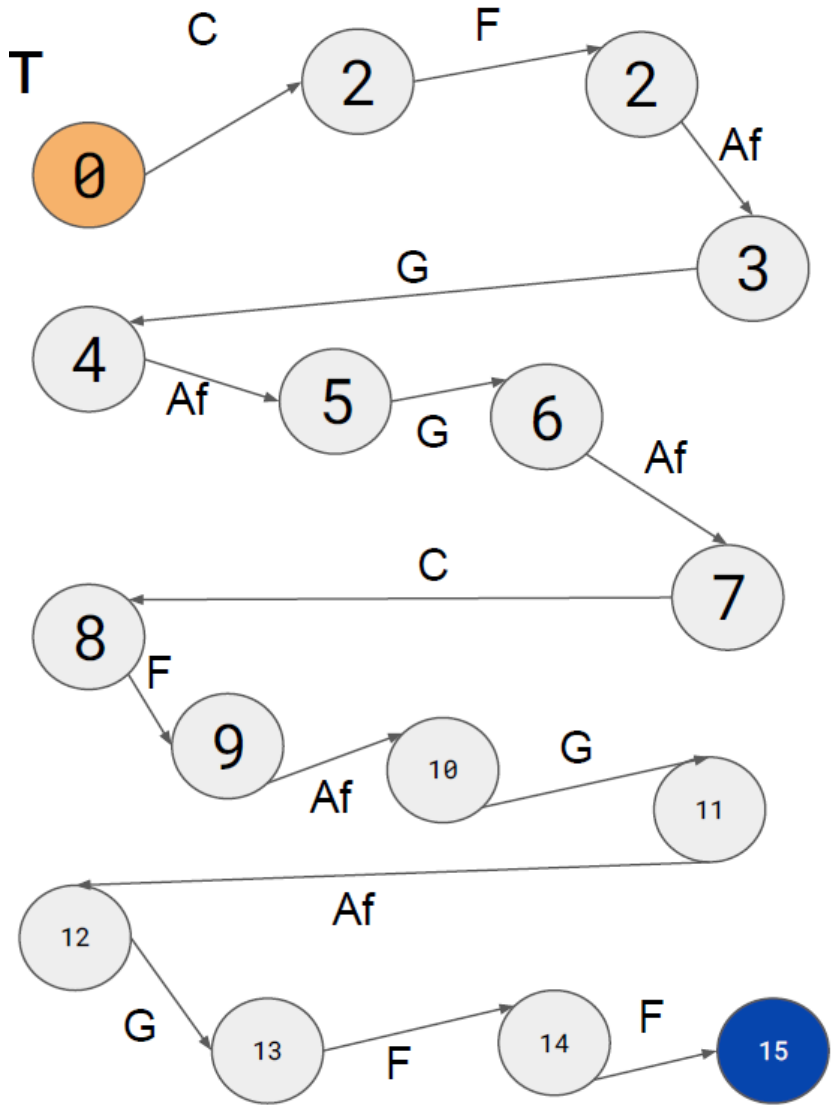
Сейчас здесь выводится "0" на все разряды ***(строчки 234-280)***

day_6/lab/top.sv*

А как распознавать не отдельные ноты, а мелодии?

Реализуем конечный автомат, распознающий последовательность

```
always @ (posedge clk or posedge reset)
    if (reset)
      states [2] <= 0;
    else
      case (states [2])
       0: if ( t_note == G  ) states [2] <=  1;
       1: if ( t_note == F  ) states [2] <=  2;
       2: if ( t_note == A  ) states [2] <=  3;
       3: if ( t_note == B  ) states [2] <=  4;
       4: if ( t_note == Cs ) states [2] <=  5;
       5: if ( t_note == D  ) states [2] <=  6;
       6: if ( t_note == E  ) states [2] <=  7;
       7: if ( t_note == F  ) states [2] <=  8;
       8: if ( t_note == E  ) states [2] <=  9;
       9: if ( t_note == D  ) states [2] <= 10;
      10: if ( t_note == C  ) states [2] <= 11;
      11: if ( t_note == Bf ) states [2] <= 12;
      12: if ( t_note == A  ) states [2] <= 13;
      13: if ( t_note == G  ) states [2] <= 14;
      14: if ( t_note == Bf ) states [2] <= recognized;
      endcase
```

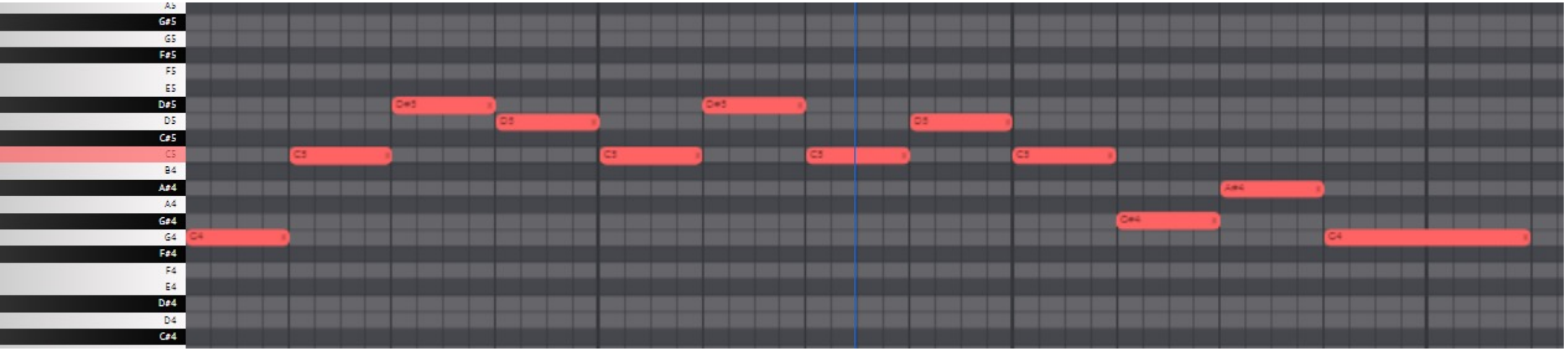


day_6/lab2/top.sv(232)

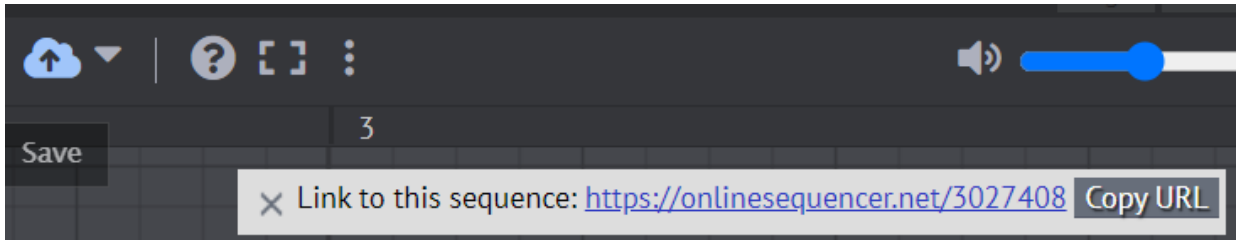
Как создать последовательность нот?

https://onlinesequencer.net

Для создания последовательности нот можно воспользоваться онлайн-секвенсором



Можно создать последовательность на пк, затем перенести ссылку на телефон и проиграть её на нём.



.mp3

Лабораторная работа №2

- **Задание 1. Распознать две заготовленные .mp3 мелодии с помощью конечного автомата:**

***Попробовать самостоятельно проиграть мелодию, которая идёт под названием simple sequence в проекте**

Можно проиграть с телефона

До	До#	Ре	Ре#	Ми	Фа	Соль	Соль#	Ля	Ля#	Си	До
C	C_s	D	D_s	E	F	G	G_s	A	A_s	B	C

- **Задание 2. Придумайте собственную последовательность нот, закодируйте её и проиграйте с помощью онлайн-секвенсора или генератора тона. Можно создать последовательность на пк, затем перенести ссылку на телефон и проиграть её на нём.**

https://onlinesequencer.net

day_6/lab/top.sv*