Laboratory exercise 1
Sine wave generator

1 Goal:

The goal is to achieve knowledge and practical experience in design of sine wave generators for modern applicatin specific computers, to get programming and debugging experience in VHDL language.

2 Theoretical information

Digital sine wave generators are widely used in DSP applications as sine, cosine wave sources for frequency conversion, discrete Fourier transform, in modems, software defined radios, radars, mobile phones, radio receivers, etc.

The following schemes of such generators are widely used:

- sine function calculation;
- lookup table-based generators;
- oscillation schemes;
- combined schemes.

Sine function calculation is the usual way to generate sine waves in PC and other program controlled computers. Here for this purpose the proper instruction of the floating point coprocessor is usually used. In the application specific processor the different approximation algorithms are used like Taylor scheme, CORDIC algorithm, interpolation algorithm, etc. For example, sine and cosine functions at the interval |x|<1 can be estimated as

$$\sin(\pi x/2) = 1{,}57063x - 0{,}64323x^3 + 0{,}07271x^5, \qquad (1)$$
$$\cos(\pi x/2) = 0{,}9994 - 1{,}22279x^2 + 0{,}22399x^4,$$

with the error, which is less than 0,06%. The disadvantage of this method consists in the large complexity of calculations (in the example above – up to 6 multiplications and 2 additions for the sine function). Besides, here the functions are defined for the angles less than $\pi/2$, and additional calculations are needed for deriving the functions in another ranges. The advantage is a wide range of the generated frequencies.

Lookup table-based generator is the simplest and most widely used solution. The idea is to build the table of M samples of the sine function, which form a single sine wave period. That means that at the address i the value $S=\sin(2\pi i/M)$ is stored. The wave generation means reading the samples, addressing them by the incremented address counter. The increment $k=1,2,...,M/2$ of such a counter is proportional to the resulting sine wave frequency $f = kf_S/M$, where f_S is the sampling (quantisation) frequency. The precisions of both the outputted sine wave and its frequency installing depend on the table volume M and the data width of the coefficients S.

Oscillation scheme is an algorithm, which generates the sine waves using some fundamental properties of transcendental functions. It is usually implemented as the solving of some difference equation. For example, the following difference equation

$$y(i) = 2\cos(b)y(i-1) - y(i-2),\ \ i=0,1,..., \qquad (2)$$

models the second order recursive digital filter at the border of amplification and excitation modes. Such a scheme generates the sine wave by the initial conditions

$$y(-1)= -\sin(b);\ \ y(-2)= -\sin(2b); \qquad (3)$$

or the cosine wave by the conditions

$$y(-1)= \cos(b);\ \ y(-2)= \cos(2b);$$

with the frequency $f = bf_S/(2\pi)$ Hz. Theoretically equation (2) represents the stable sine wave generator, i.e. it operates without damping or saturation of oscillations, if the multiplier of $y(i-2)$ (if any) is equal precisely to a 1

[1], which is usually achieved without complications. But the sine and cosine coefficients must be truncated by the machine representation in such a way, that the sine of a zero angle to be equal to a zero. i.e.

$$y(0) = 2\cos(b)(-\sin(b)) - (-\sin(2b)) = 0. \quad (4)$$

On the Fig.1 the dataflow graph is represented of the equation (2) solving.

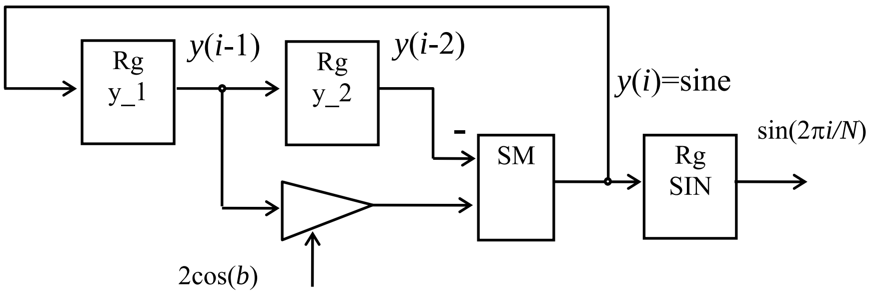

Fig.1. Dataflow graph of the equation (2) solving

This graph can be described by the following VHDL program.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity SINE is
        port (CLK: in STD_LOGIC;
                RESET: in STD_LOGIC;
                SIN: out integer range -2**15 to 2**15-1);
end SINE;

architecture SIMPLE of SINE is
signal y_1, y_2 : integer range -2**15 to 2**15-1;
constant   sinb: integer :=-286;
constant   sin2b: integer :=-572;
constant   cosb: integer :=32767;
begin
         process(CLK,RESET)
        variable sine: integer;
                begin
                if RESET ='1' then
                        y_1<=sinb;
                        y_2<=sin2b;
                        SIN<=0;
                elsif CLK='1' and CLK'event then
                        sine:=2*cosb*y_1 /2**15
                        y_2<=y_1;
                        y_1<=sine;
                        SIN<=sine;
                end if;
        end process;
end SIMPLE;
```

The sine function as well as the usual signal data have the representation range from -1.0 to 1.0. But in VHDL the signals are represented by integers or bit vectors. Therefore, in this model all the values are to be scaled with the coefficient 2^{-15}. Initial data of the signals `y1`, `y2` and constants `cosb` = $2^{15}\cos(b)$ for the different frequencies are shown in the following table 1.

Table 1

-sinb	-sin2b	cosb	Calculated sine wave period, clocks	Derived sine wave period, clocks	Derived magnitude	DC dis-placement
19261	31165	26510	10	10	32767	0
2856	5690	32642	72	72	32422	-6
286	572	32767	720	703	28808	-87

This generator is rather simple. It is often used to generate the waves of a single frequency. The disadvantages of this generator are: -small scope of the frequency regulating, which is limited by the data bit width and relation (4); - wave magnitude is different for the different frequencies, and some displacement is present (see the table); - for a set of frequencies the set of coefficients must be calculated and/or stored; - very high ($>0{,}13f_S$) and very low ($<0{,}001f_S$) frequencies could not be generated without extreme errors. A

Another oscillation schema is based on the well-known trigonometric formulas:

$$\sin(x+y) = \sin x \cos y + \cos x \sin y; \qquad (5)$$
$$\cos(x+y) = \cos x \cos y - \sin x \sin y.$$

Here sin$x$, cos$x$ are samples of the generated waves, and y is the angle to which the neighboring samples are differ, i.e. it represents the given frequency. The disadvantage of this sheme consists in its unstability due to the unprecise representation and calculation of the sine and cosine samples. That means that $\sin^2 x + \cos^2 x \neq 1$ $\sin^2 y + \cos^2 y \neq 1$ due to the truncation errors (2) . This feature can be minimized by addition of some nonlinearities to this schema which will decrease the increased signal magnitude.

In the combined schemes the superposition of the mentioned above schemes is used. For example, consider the generator which frequency must be tuned precisely. Then such generator can be built as two generators, one of them generates sine and cosine waves with the high frequency and another one does them with the low frequency. The resulting signal is derived by the mixing the signals of both of them using the equations (5).

3. Generator design example

Consider the design of the sine wave generator, which outputs the sine, cosine waves. Its parameters are: frequency $0{,}01f_S$, sine table length 16, table pattern – sine function of the argument range 0 to π, phase accumulator data width – 16 bit, output data width - 16.

The generator structure is shown on the Fig.2

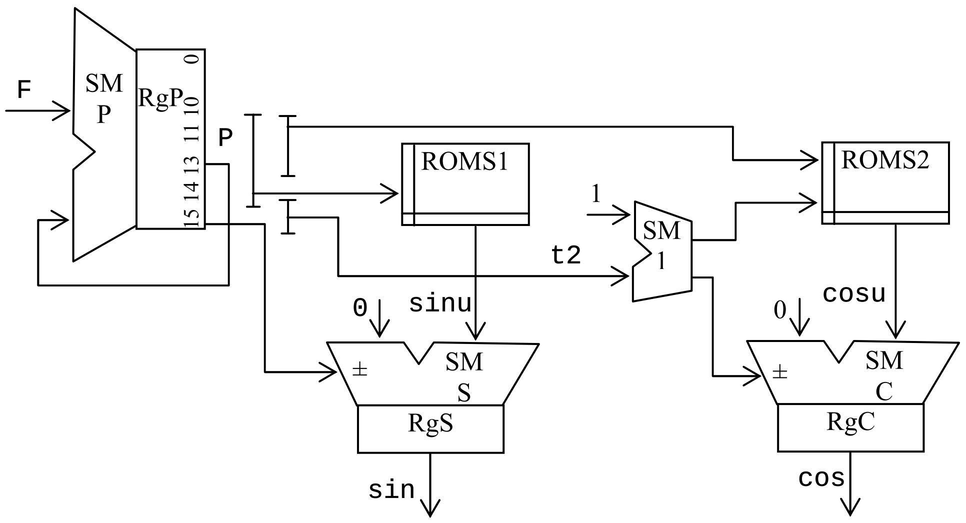

Fig.2. Sine wave generator structure

The adder SMP with the register RgP implements the phase accumulator with the increment F. ROM ROMS1, ROMS2 store one half of the sine function period. ROMS1 is addressed by 5 most significant bits of RgP except highest one. Therefore, the whole sine wave period consists of 32 samples.

The adder SMS inverts the sine code to generate the negative waves of the resulting sine signal. It is implemented when the 15-th bit of RgP is a 1. In another situation this adder throughputs the data without exchanges.

The adder SM1 adds a 1 to 2 MSBs of the code RgP, and therefore, it shifts the phase value to 90°. By this method the address is derived which provides the cosine function fetching from the sine table. The adder SMC and ROMS2 generate the cosine function. Sine and cosine samples are buffered in the registers RgS and RgC respectively.

The end of each wave period is in time with the phase accumulator overflow. This means that to generate the signal with the frequency $0{,}01 f_S$, 100 clock cycles must take one overflow. Such situation occurs when the phase increment is equal to $F =]2^{16}/100[= 655$. The relative frequency error due to the phase code truncation is equal to $\delta f = (2^{16}/100 - F)/F = 5{,}5\cdot 10^{-4}$.

This generator is described as the following.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_signed.all;
entity SIN_GEN is
  port(CLK : in STD_LOGIC;
          RST : in STD_LOGIC;
          F : in STD_LOGIC_VECTOR(15 downto 0);
          SIN_O : out STD_LOGIC_VECTOR(15 downto 0);
          COS_O : out STD_LOGIC_VECTOR(15 downto 0));
end SIN_GEN;
architecture beh of SIN_GEN is
  type ARR is array (0 to 15) of integer range -2**15 to 2**15-1;
  constant SIN_TABLE:ARR:=
          (0,      6392,12539,18204, 23170,27245,30273,32138,
          32767,32138,30273,27245, 23170,18204,12539,6392);
  signal t:ARR:= (0,others=>0);
  signal P,sinu,cosu:STD_LOGIC_VECTOR(15 downto 0);
  signal t2:STD_LOGIC_VECTOR(1 downto 0);
begin
  process
          variable j: real;
  begin
          for i in 0 to 15 loop
                  j:=real(i);
                  t(i)<=integer(FLOOR(32767.9* SIN(MATH_2_PI*j/32.0)));
          end loop;
          wait;
  end process;

  CT_PHASE:process(CLK,RST) begin     --счетчик фазы
          if RST='1' then
                  P<=X"0000";
          elsif CLK='1' and CLK'event then
                  P<=P+F;
          end if;
  end process;

  sinu<=CONV_STD_LOGIC_VECTOR(SIN_TABLE(CONV_INTEGER('0'&P(14 downto 11))),16);

  t2<=P(15 downto 14) +1;
 cosu<=CONV_STD_LOGIC_VECTOR(SIN_TABLE(CONV_INTEGER('0'&t2(0)&P(13 downto 11))),16);

  process(CLK,RST) begin          -- registers with 2th complement deriving
          if RST='1' then
                  SIN_O<=(others=>'0');
```

```
                COS_O<=(others=>'0');
        elsif CLK='1' and CLK'event then
                if      P(15)='0' then
                        SIN_O<= sinu;
                else
                        SIN_O<=0 - sinu;
                end if;
                if      t2(1)='0' then
                        COS_O<= cosu;
                else
                        COS_O<=0 - cosu;
                end if;
        end if;
    end process;
end beh;
```

4. **Laboratory exercise implementation**

The generator must generate both sine and cosine waves. The output signal magnitude must be stable. The generator must be able to generate a set of frequencies f_1, f_2 , and f_3 , which are given by the 2-bit input code. The inputs and outputs must be represented by the STD_LOGIC codes.

Each exercise variant has a set of parameters, which are numbered by natural numbers. A set of them is derived from the record-book number of the student. Consider 3 last figures a_2,a_1,a_0, of the record-book number. Then the variant number is

$N = 100a_2 + 10a_1 + a_0 = 2^9b_9+2^8b_8+2^7b_7+2^6b_6+2^5b_5+2^4b_4+2^3b_3+2^2b_2+2^1b_9+b_0$,

where b_i are the bits of the number N in the binary representation.

The generator type is selected from the Table 1. .

Table 1

b_0	0	1
Type	Lookup table-based	Oscillator scheme (2)

The set of calculated frequencies is represented in the Table 2.

Table 2

b_3,b_2, b_1	Generator frequencies, fractions of f_C		
000	0,001	0,01	0,03
001	0,002	0,02	0,04
010	0,003	0,03	0,05
011	0,004	0,04	0,06
100	0,005	0,05	0,07
101	0,006	0,06	0,08
110	0,007	0,07	0,09
111	0,008	0,08	0,1

The generator must have the bit widths as in the following table

Table 3

b_4,b_3, b_2	Output sine bit width	Phase accumu-lator bit width	Coefficient, ALU bit widths	Sine table address bit width
000	8	16	12	4
001	8	18	14	4
010	10	20	16	5
011	10	22	18	5
100	12	24	20	5
101	12	26	22	6
110	16	28	24	6
111	16	32	28	6

5.Testbench development and model testing

The model can be tested in the ActiveHDL environment by adding the stimulator signals for CLK and RST inputs, and investigating the output sine and cosine signals. The CLK frequency must be equal to 100 MHz.

For each given frequency code the output sine wave frequency must be measured by the invesitigation of the output waveform diagram. The waveform frequency error must be derived too.

6. Laboratory exercise report

The laboratory exercise report must contain:

- Goal of the work,
- Generator description,
- VHDL texts,
- Waveforms of testing,
- Conclusions.

Literature

1. Отнес Р., Эноксон Л. Прикладной анализ временных рядов. –М.:Мир. –1982. – 428 с.
2. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов. –М.:Мир. –1978. – 848 с.