# VHDL Implementation of Direct Digital Synthesis

Tristan Itschner

June 4, 2022

## Contents



## 1 Introduction

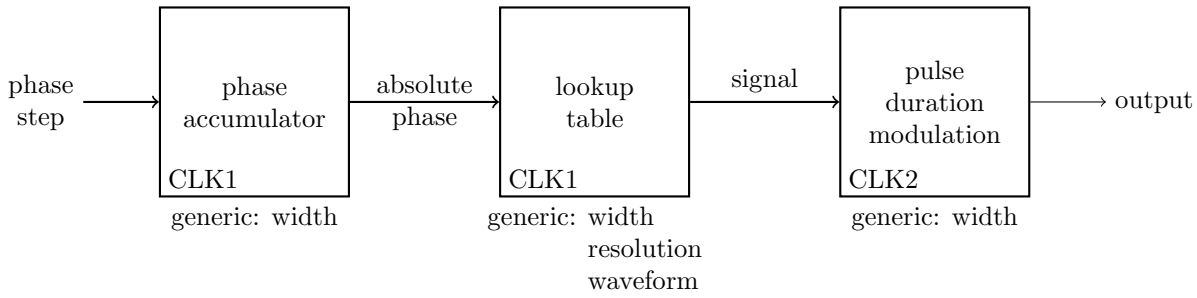


Figure 1: High-Level DDS Block Diagram (PL domain only)

Figure 1 shows the high level diagram of Direct Digital Synthesis.

Direct Digital Synthesis (DDS) is a digital method of periodic signal generation that has almost universally replaced prior analog methods.

# 2 Components

## 2.1 Phase Accumulator

The phase accumulator is nothing more than an adder, that continually adds the input signal to its internal register, which is at the same time the output.

Common values of the registers width are 48 bits.

```vhdl
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;

entity phase_acc is
  generic (
            width : natural := 48
          );
  port                  (
         clk  : in std_logic;
         rstn : in std_logic; -- not used, may be used, but not necessarily so
         sel_phasestep : in  std_logic_vector(width - 1 downto 0);
         ph_o          : out std_logic_vector(width - 1 downto 0)
       );
begin
end;

architecture a_phase_acc of phase_acc is
  signal step  : unsigned(sel_phasestep'range);
  signal phase : unsigned(ph_o'range) := (others => '0');
begin

  step  <= unsigned(sel_phasestep);
  ph_o  <= std_logic_vector(phase);
  phase <= phase + step when rising_edge(clk);

end;
```

../phase_acc.vhd

## 2.2 Lookup Table

The lookup table converts the "time value" that is provided by the phase accumulator into the corresponding signal value.

It is a block ram with usually one cycle of delay.

This block always runs at the same clock as the phase accumulator.

Usually only the MSbs of the input signal are relevant. This is due to the cost of a large block ram. Still, the additional bits in the phase accumulator provide better time resolution. They may also be used for interpolation methods, to increase the signal resolution.

Common values are 16 bits, which is the likewise the limit for audio perception.

```vhdl
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;

use ieee.math_real.all;

entity lookup_table is
  generic (
```

```
            input_width   : natural := 48;
            output_width : natural := 8
          );
  port (
         clk    : in  std_logic;
         phase : in  std_logic_vector(input_width - 1 downto 0);
         sig_o : out std_logic_vector(output_width - 1 downto 0)
       );
end;

architecture a_lookup_table of lookup_table is
  type rom is array (integer range <>) of signed(output_width - 1 downto 0);

  impure function fillrom return rom is
    variable t : rom(2**input_width - 1 downto 0);
  begin
    for i in t'range loop
      --t(i) := to_unsigned(natural((((real(max_phase)/2.0) -
    1.0))*sin(MATH_2_PI*real(i)/real(max_phase))
      --+ (real(max_phase)/2.0 - 1.0)),t(i)'length);
      t(i) := to_signed(integer((((real(2**output_width)/2.0) -
    1.0))*sin(MATH_PI*real(i)/real(2*2**output_width))), t(i)'length);
    end loop;
    return t;
  end function;

  constant table : rom(2**input_width - 1 downto 0) := fillrom;
  attribute ram_style : string;
  attribute ram_style of table : constant is "block";
begin

  sig_o <= std_logic_vector(table(to_integer(unsigned(phase)))) when
    rising_edge(clk);

end;
```

../lookup_table.vhd

## 2.3 Pulse Duration Modulation

Pulse Duration Modulation is a method of converting a digital signal into an analog signal. It is similar to delta signal modulation (if not identical). The PDM block is only one part, on the other end there must by an analogy reconstruction filter.

The PDM utilizes a counter that is increased by one every cycle and has a width that must be identical to the input signal resolution. The output signal is merely 1 bit, and it is 1 if the signal value is above the counter value, else it's 0.

The PDM block may run at a faster frequency than the input signal. This allows for a better signal reconstruction, as the edge frequency of the analog recontruction filter may be choosen higher. Also a slower frequency on the block ram side allows for a larger block ram.

```
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;

entity pdm is
  generic (
            width : natural := 8
```

```
          );
  port (
         clk : in  std_logic;
         rst : in  std_logic;
         x   : in  std_logic_vector(width - 1 downto 0);
         y   : out std_logic
       -- other ports
       );
end;

architecture a_pdm of pdm is
  signal counter    : signed(x'range);
begin

  y <= '1' when signed(x) > counter else '0';

  process (all) is
  begin
    if rising_edge(clk) then
      counter <= counter + "01"; -- vhdl is very stupid...
      if rst then
        counter <= (others => '0');
      end if;
    end if;
  end process;


end;
```

../pdm.vhd

## 2.4 DDS Wrapper

```
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;

entity dds is
  generic (
            acc_width         : natural := 8;
            sig_width         : natural := 8;
            log_clock_divider : natural := 8
          );
  port (
         clk           : in  std_logic;
         rst           : in  std_logic;
         sel_phasestep : in  std_logic_vector(acc_width - 1 downto 0);
         y             : out std_logic
       );
end;

architecture a_dds of dds is
  signal ph      : std_logic_vector (acc_width   - 1 downto 0);
  signal sig     : std_logic_vector (sig_width   - 1 downto 0);
  signal clk_div : unsigned         (log_clock_divider - 1 downto 0);
  signal clk_2   : std_logic;
begin

  process (all) is
```

```
27  begin
28    if rising_edge(clk) then
29      clk_div <= clk_div + "1";
30      if rst then
31        clk_div <= (others => '0');
32      end if;
33    end if;
34  end process;
35  clk_2 <= not clk_div(clk_div'high);
36
37  phase_acc_inst: entity work.phase_acc
38  generic map (
39                width     => acc_width
40              )
41  port map (
42              clk           => clk_2,
43              rstn          => not rst,
44              sel_phasestep => sel_phasestep,
45              ph_o          => ph
46            );
47
48  lookup_table_inst : entity work.lookup_table
49  generic map (
50                input_width  => acc_width,
51                output_width => sig_width
52              )
53  port map (
54              clk   => clk_2,
55              phase => ph,
56              sig_o => sig
57            );
58
59
60  pdm_inst : entity work.pdm
61  generic map (
62                width => sig_width
63              )
64  port map (
65              clk => clk,
66              rst => rst,
67              x   => sig,
68              y   => y
69            );
70
71  end;
```

../dds.vhd

### 2.5 FAQ

- Why is the phase accumulator's register width greater than the lookup table?

## 3 Testbench