MS3: Direct Digital Synthesis

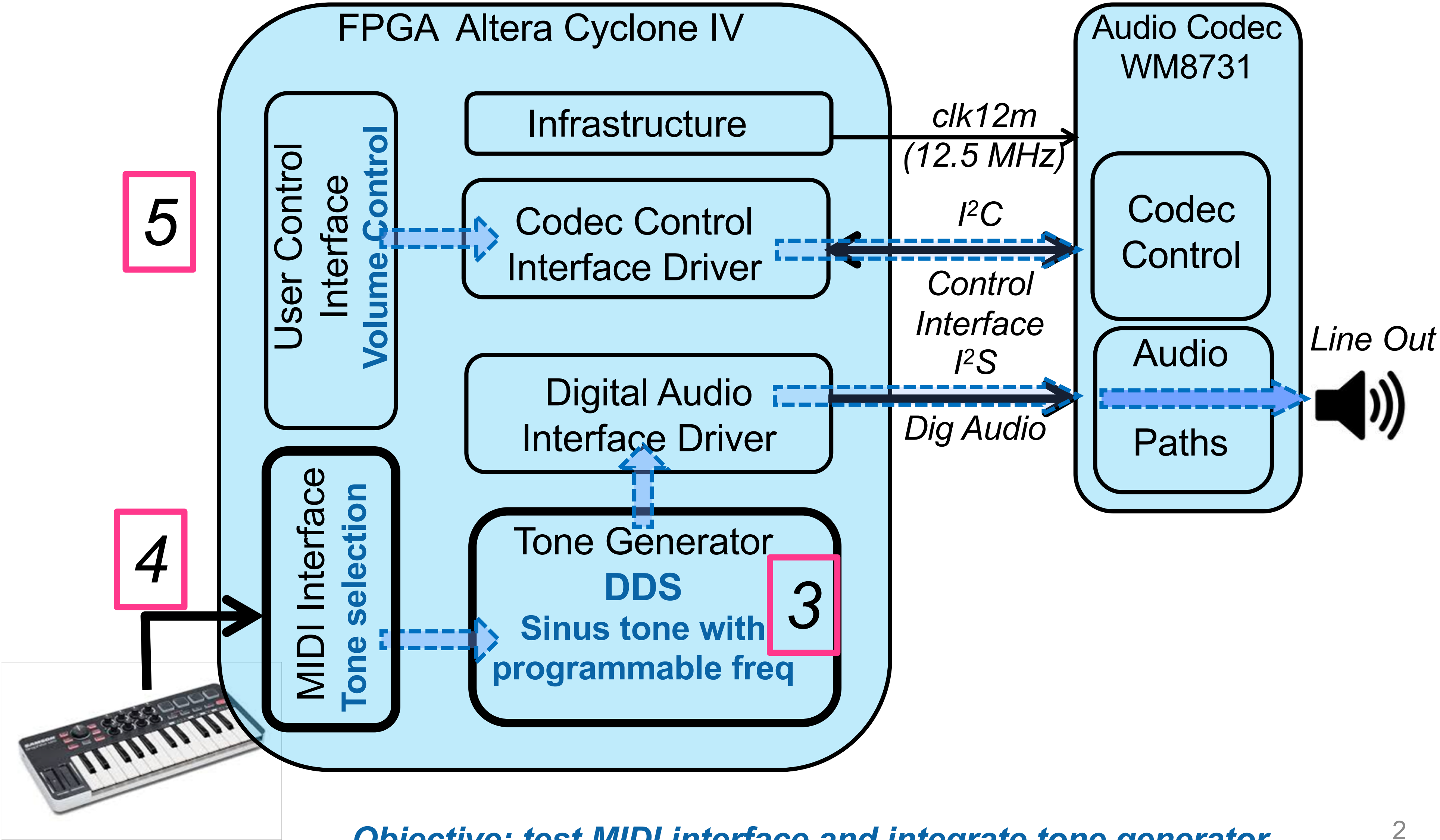
School of Engineering

Direct Digital Synthesis (DDS)

- Block Partitioning
- Basic Principle
- Implementation Example
- Counter-Length and Increment Calculation
- VHDL Implementation
- MIDI Integration

Milestones 3 bis 5
Tone Generator und MIDI-Interface

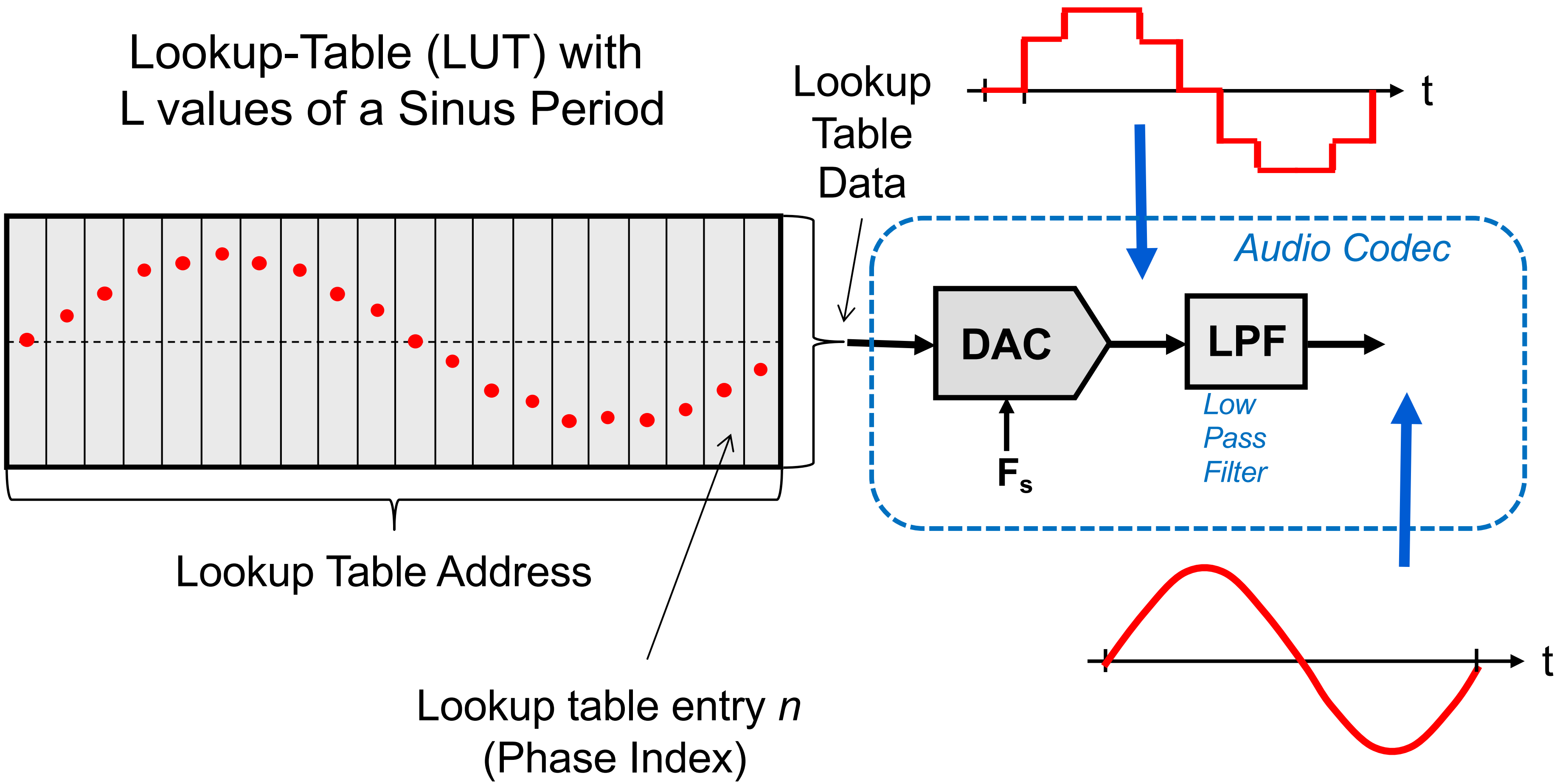
School of Engineering



Objective: test MIDI interface and integrate tone generator

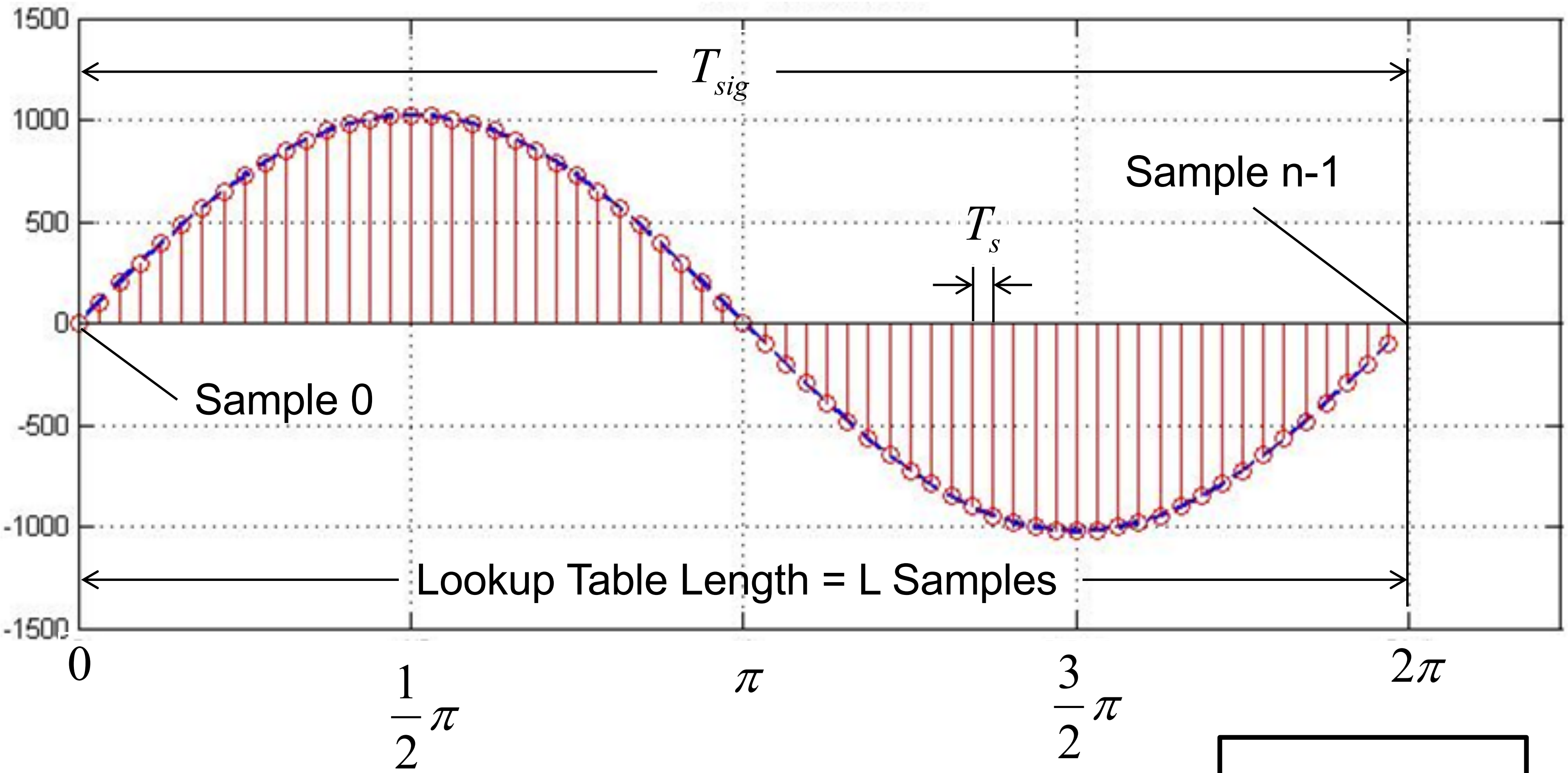
DDS: Basic Principle

School of Engineering



Frequency of a synthesized Sinus

School of Engineering



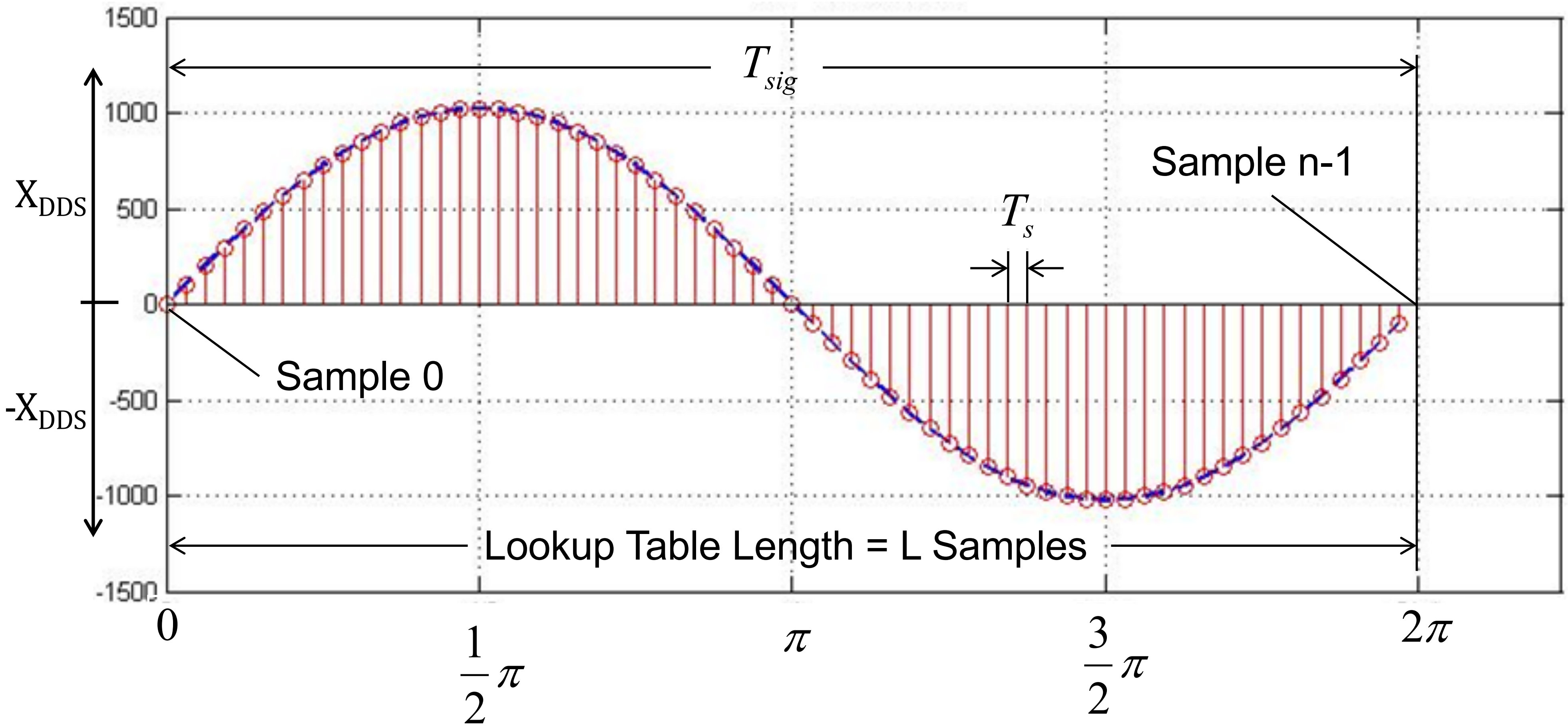
f_s= audio sample frequency
L = length of LUT
n = index for phase counter
f_{sig} = signal frequency

$$f_{sig} = \frac{1}{T_{sig}} \qquad T_{sig} = L \cdot T_s$$

$$\boxed{f_{sig} = \frac{fs}{L}}$$

Amplitude of a Synthesized Sinus

School of Engineering

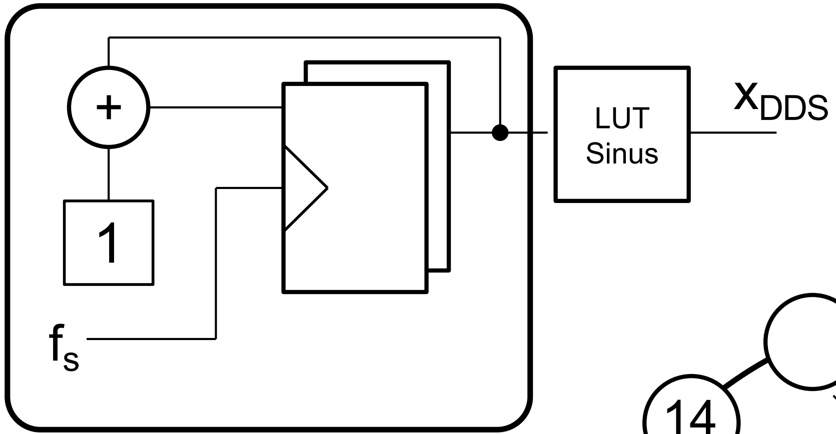


L = length of LUT
n = index of sample
A = max. amplitude

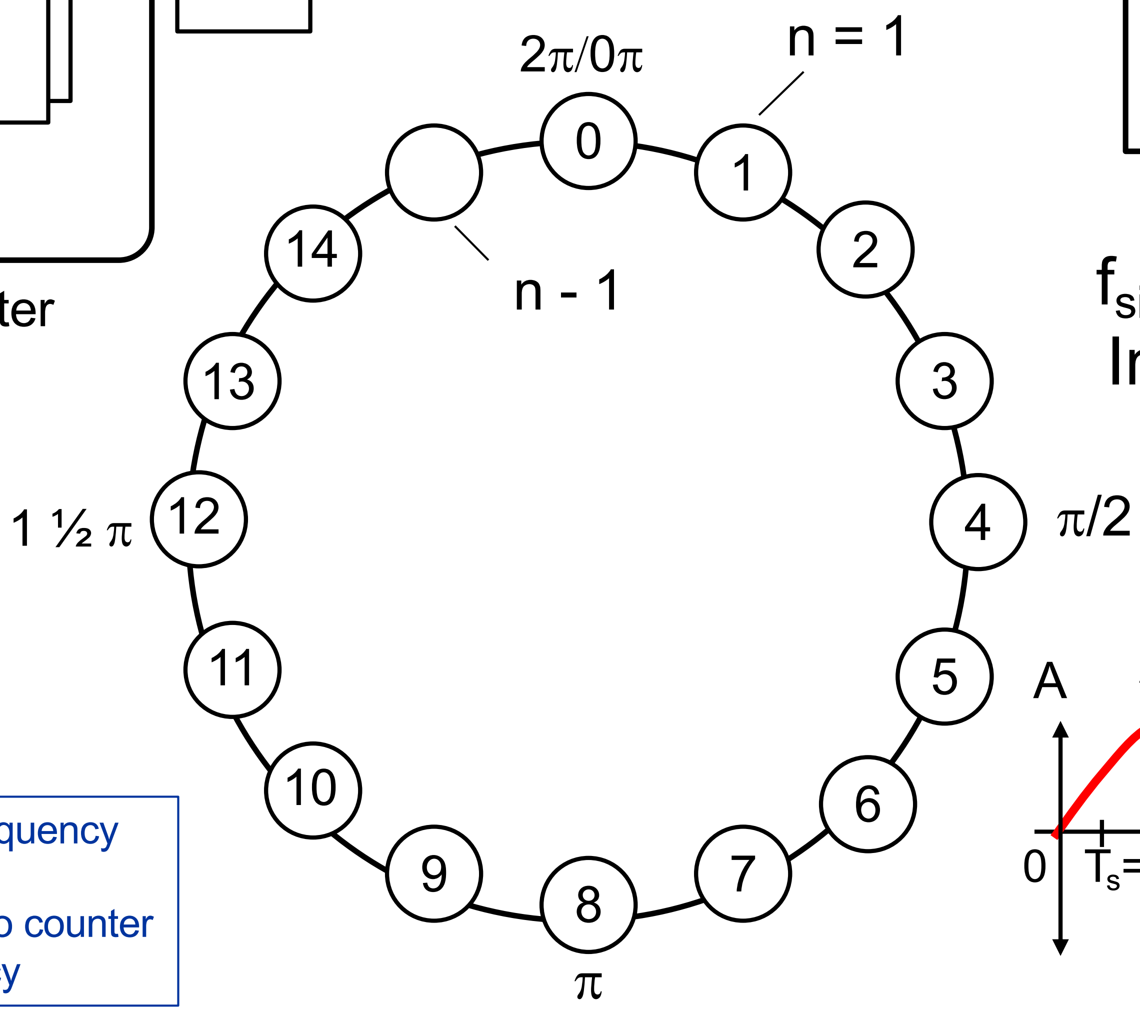
$$x_{DDS}[n] = A \cdot \sin\left(n \cdot \frac{2\pi}{L}\right)$$

LUT Adress Generation with Modulo Counter

School of Engineering

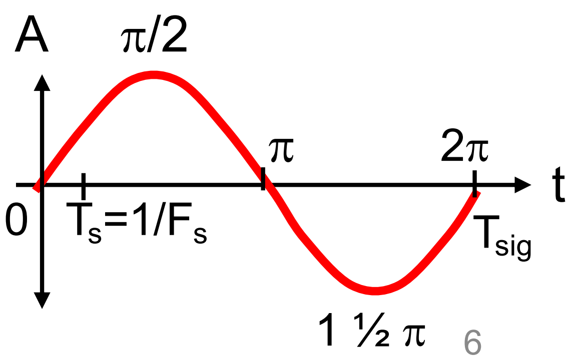


Phase Counter



$$f_{sig} = \frac{fs}{L}$$

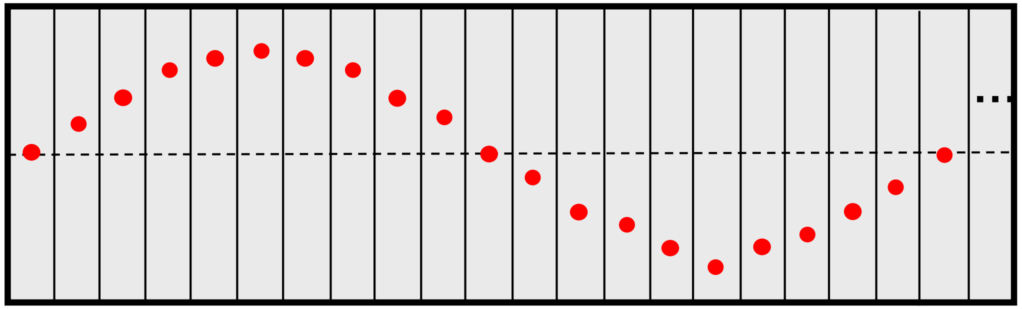
f_{sig} for counter Increment = 1



f_s= audio sample frequency
L = length of LUT
n = index for modulo counter
f_{sig} = signal frequency

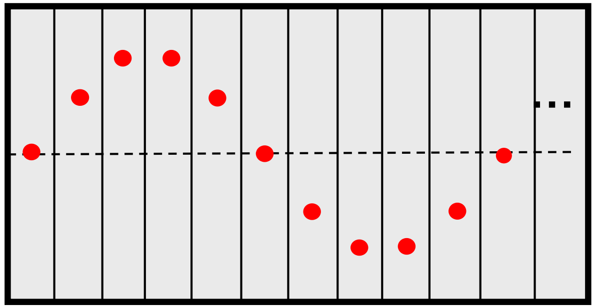
Example when LUT Length L=20

using M=Δn=1



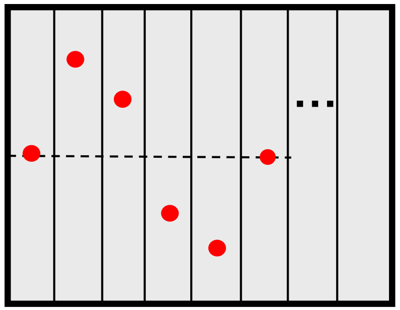
Lookup Table Address n = 0 1 2 3 4 … 19 0 …

using M=Δn=2



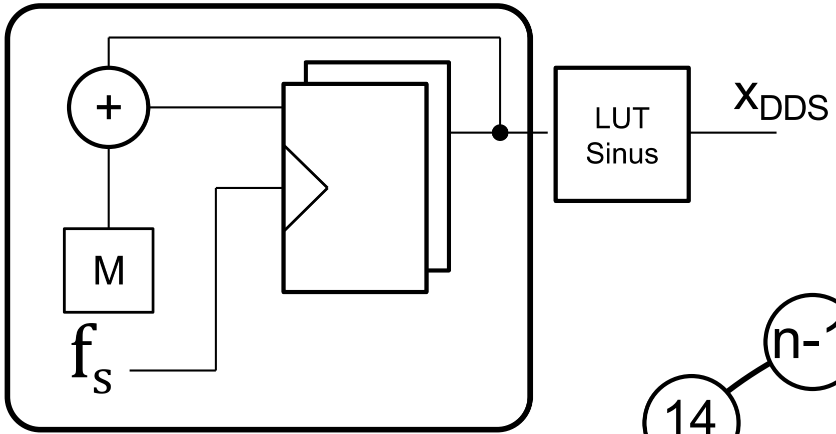
Lookup Table Address n = 0 2 4 6 4 18 0 …

using M=Δn=4

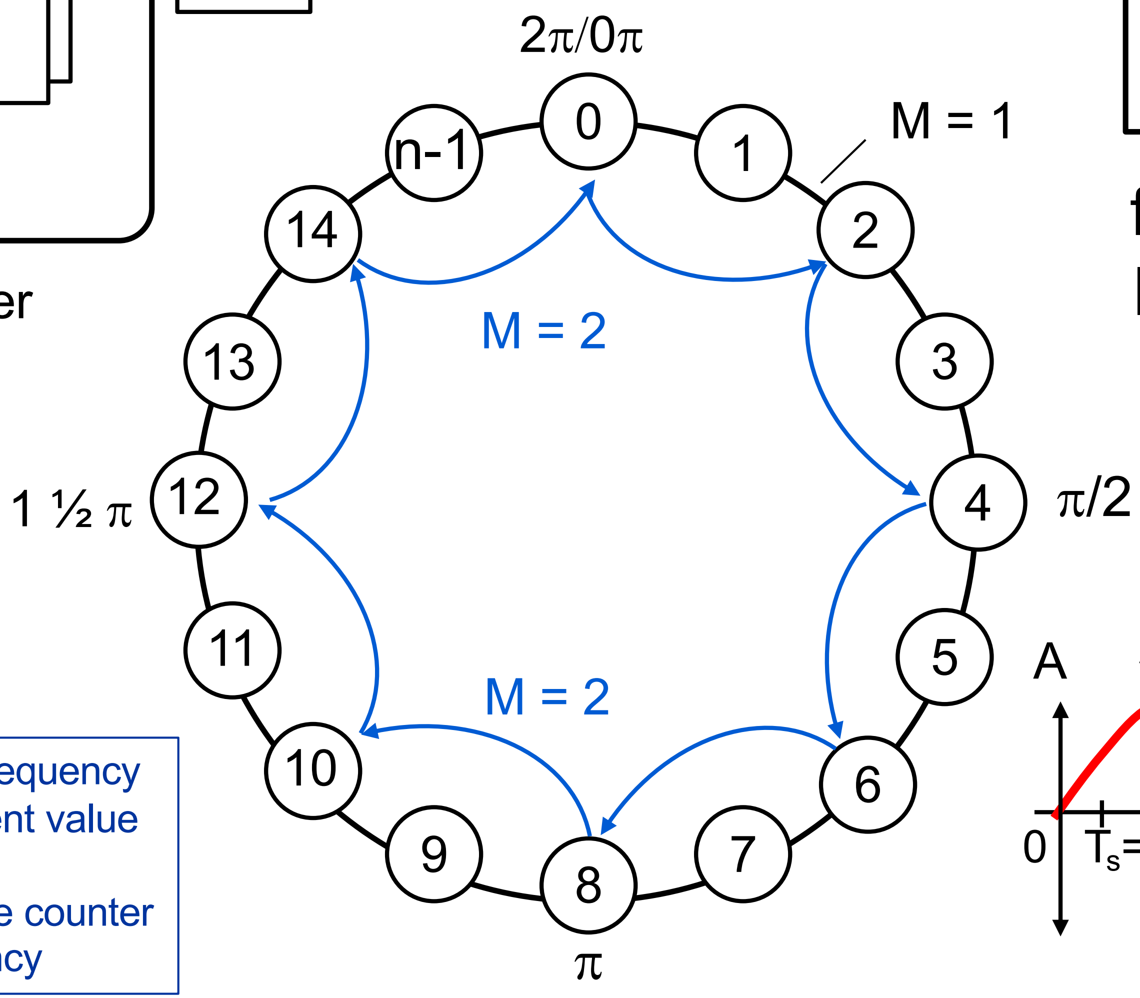


Lookup Table Address n = 0 4 8 12 16 0 …

Controlling Frequency of f_{sig}

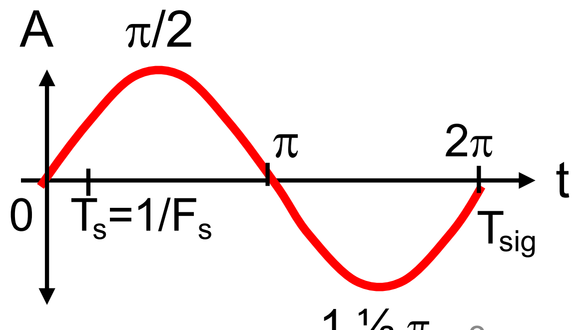


Phase Counter



$$f_{sig} = \frac{f_S}{L} M$$

f_{sig} for counter Increment = M



f_s= audio sample frequency
M = phase increment value
L = length of LUT
n = index for phase counter
f_{sig} = signal frequency

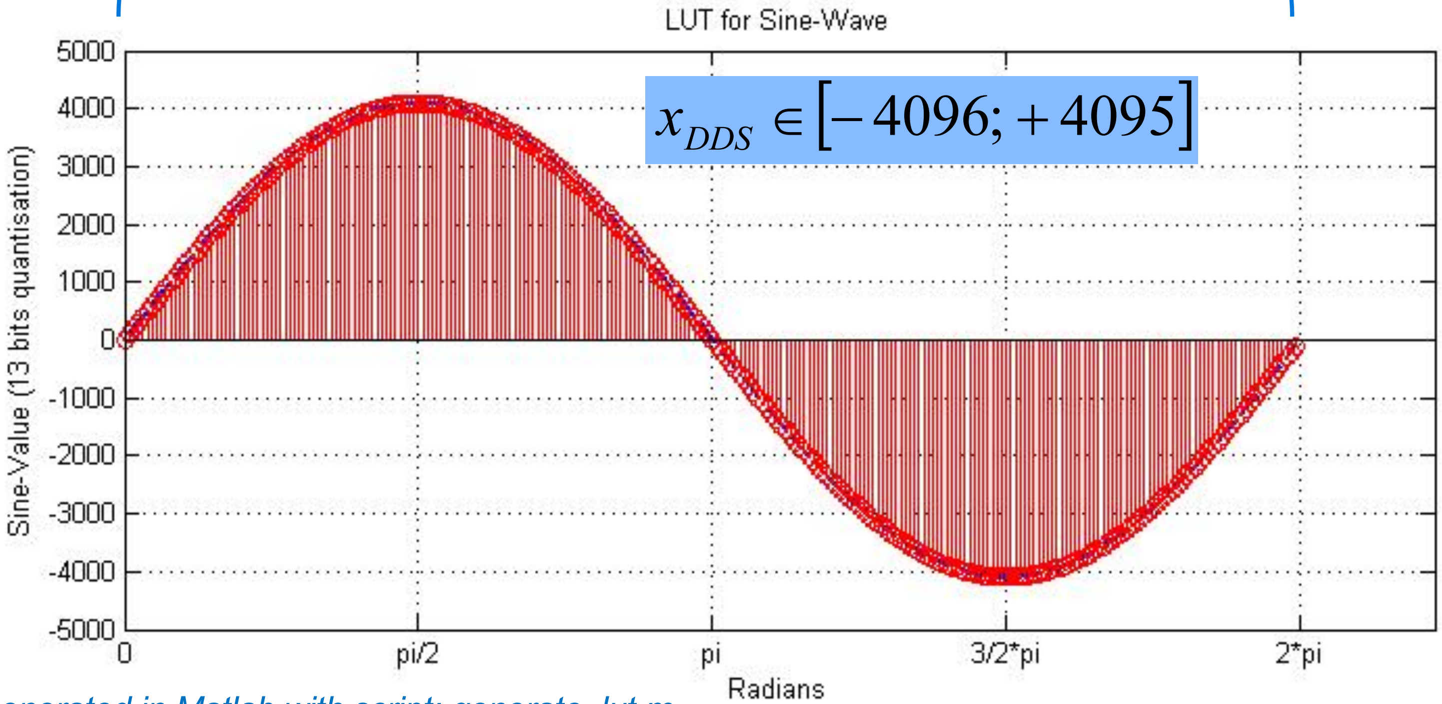
Proposal for LUT Implementation

School of Engineering

N_LUT = 8-bits (lut_sinus length L = 256 entries)
N_RESOL = 13-bits (signed = -4096..+4095)

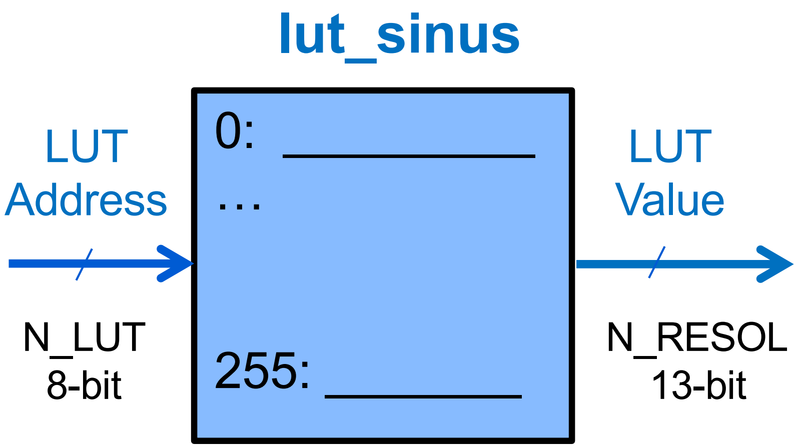
N_LUT

N_RESOL



Generated in Matlab with script: generate_lut.m

Required ROM size



Synthesised Tone

(2'complement values with 13-Bits)

$$= A \cdot \sin\left(n \cdot M \cdot \frac{2\pi}{L} \right) =$$

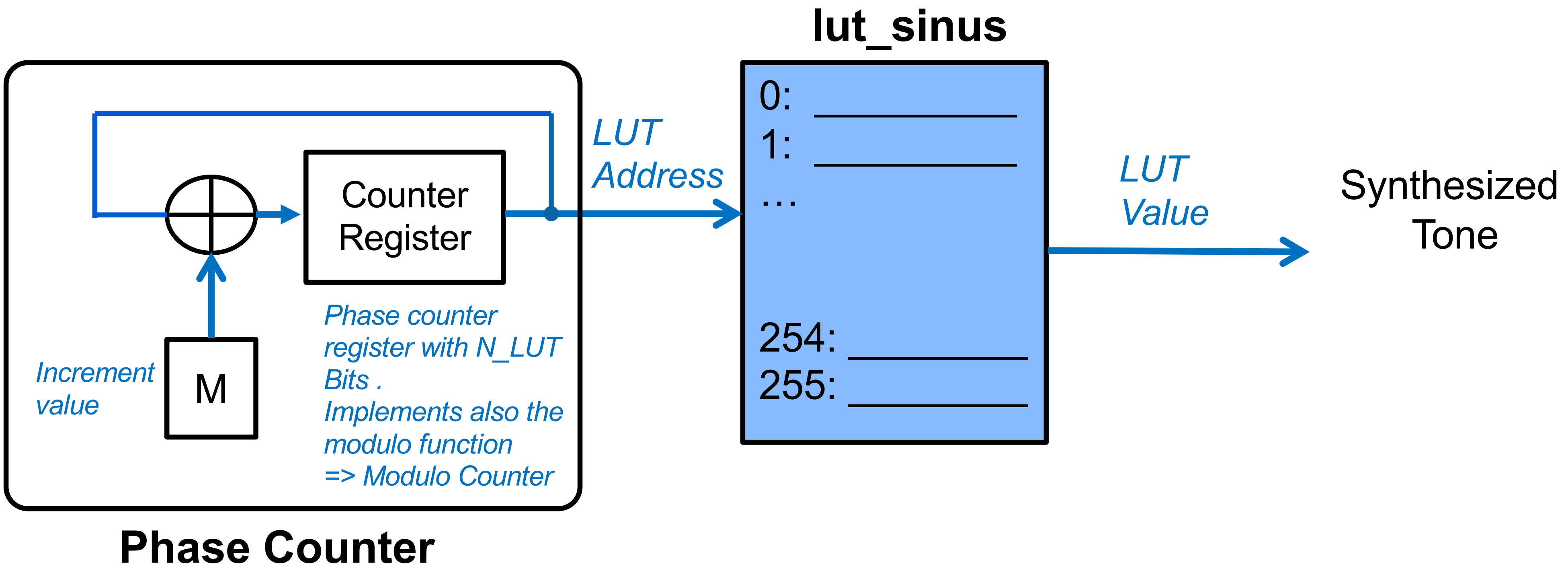
$$4096 \cdot \sin\left(n \cdot M \cdot \frac{2\pi}{256} \right)$$

$$LUT\_size = 2^8 \cdot 13 = 3'328\ bits$$

Available Memory Bits in 4CE-115 FPGA = 3'888'000 bits

Resulting Frequency Steps
when N_LUT = 8 bits

School of Engineering



Which frequency is synthesised with update on phase counter every F_s= 48kHz and:

$$f_{sig} = F_s \cdot \frac{M}{L} = 48k \cdot \frac{M}{256}$$

- M=1 … $\Rightarrow f_{sig} = 187{,}5Hz$
- M=2 … $\Rightarrow f_{sig} = 375Hz$
- M=3 … $\Rightarrow f_{sig} = 562{,}5Hz$

Keyboard Tasten vs. Frequenzen

School of Engineering

```
-- Piano Mid-Octave (white keys)
----------------------------------------------
-- DO-C4  tone ~261.63Hz
-- RE_D4  tone ~293.66Hz
-- MI_E4  tone ~329.63Hz
-- FA_F4  tone ~349.23Hz
-- SOL_G4 tone ~392.00Hz
-- LA_A4  tone ~440.00Hz
-- SI_B4  tone ~493.88Hz
-- DO_C5  tone ~523.25Hz

-- Piano Mid-Octave (black keys)
----------------------------------------------
-- DOS_C4S  tone ~277.18Hz
-- RES_D4S  tone ~311.13Hz
-- FAS_F4S  tone ~369.99Hz
-- SOLS_G4S tone ~415.30Hz
-- LAS_A4S  tone ~466.16Hz
```

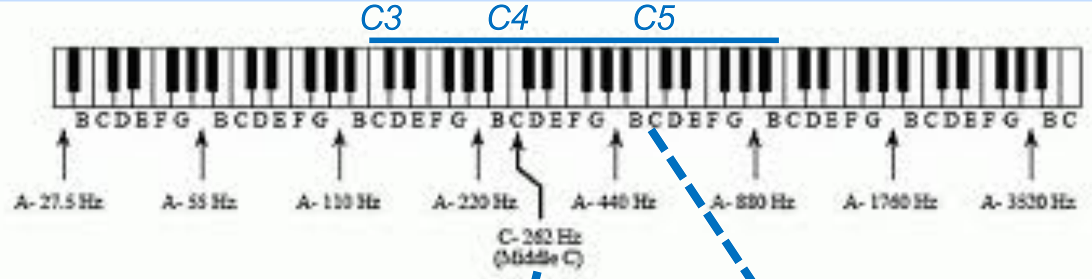
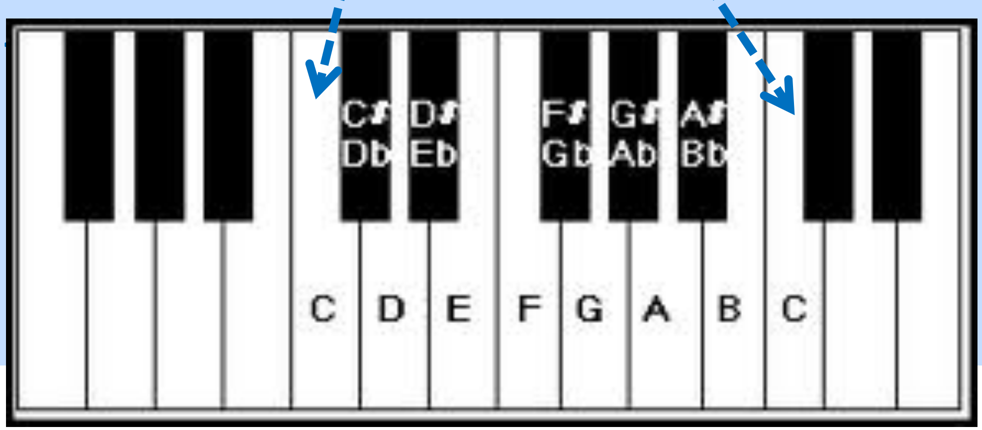


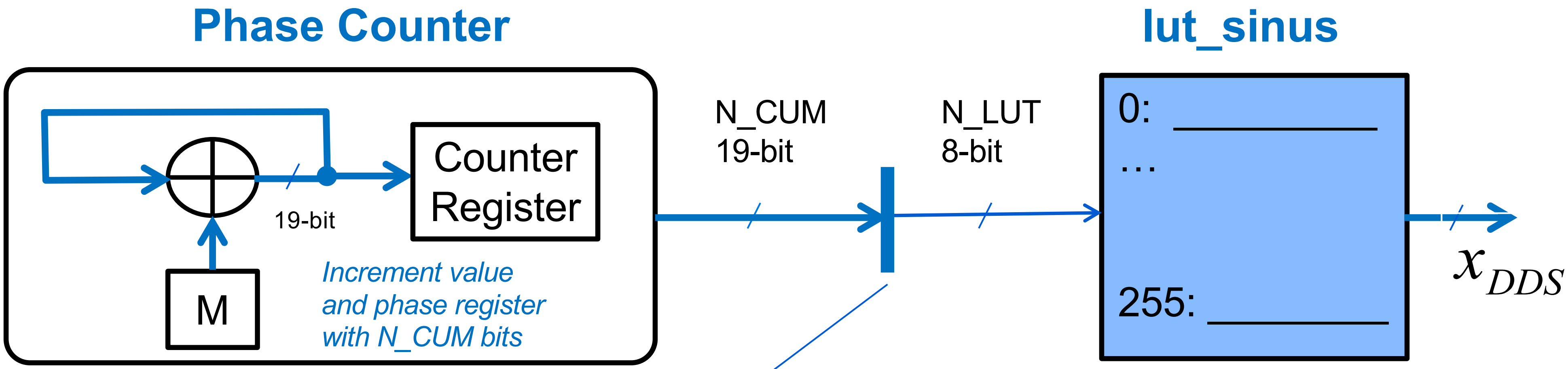
FIGURE 22-4
The Piano keyboard. The keyboard of the piano is a *logarithmic* frequency scale, with the fundamental frequency doubling after every seven white keys. These white keys are the notes: *A, B, C, D, E, F and G.*



Making Counter Bits > N_LUT

School of Engineering

Obs.: this means taking a fractional value for the increment!



Truncation:
Take only most significant 8-bits of N_CUM value for LUT Address

$$\Delta f = \frac{F_s}{2^{N\_CUM}} = \frac{48k}{2^{19}} = 0{,}0916Hz$$

0.0916 Hz is precision of tone setting

Frequency steps when N_LUT = 19 bits

School of Engineering

Which frequency is synthesised with update on phase counter every f_s= 48kHz and N_LUT = 19 bits?

$$f_{sig} = f_s \cdot \frac{M}{L} = 48k \cdot \frac{M}{2^{19}}$$

- M=1 ... $\Rightarrow f_{sig} = 0{,}092 Hz$
- M=2 ... $\Rightarrow f_{sig} = 0.183 Hz$

$$\boxed{M = \frac{f_{sig} * L}{f_s}}$$

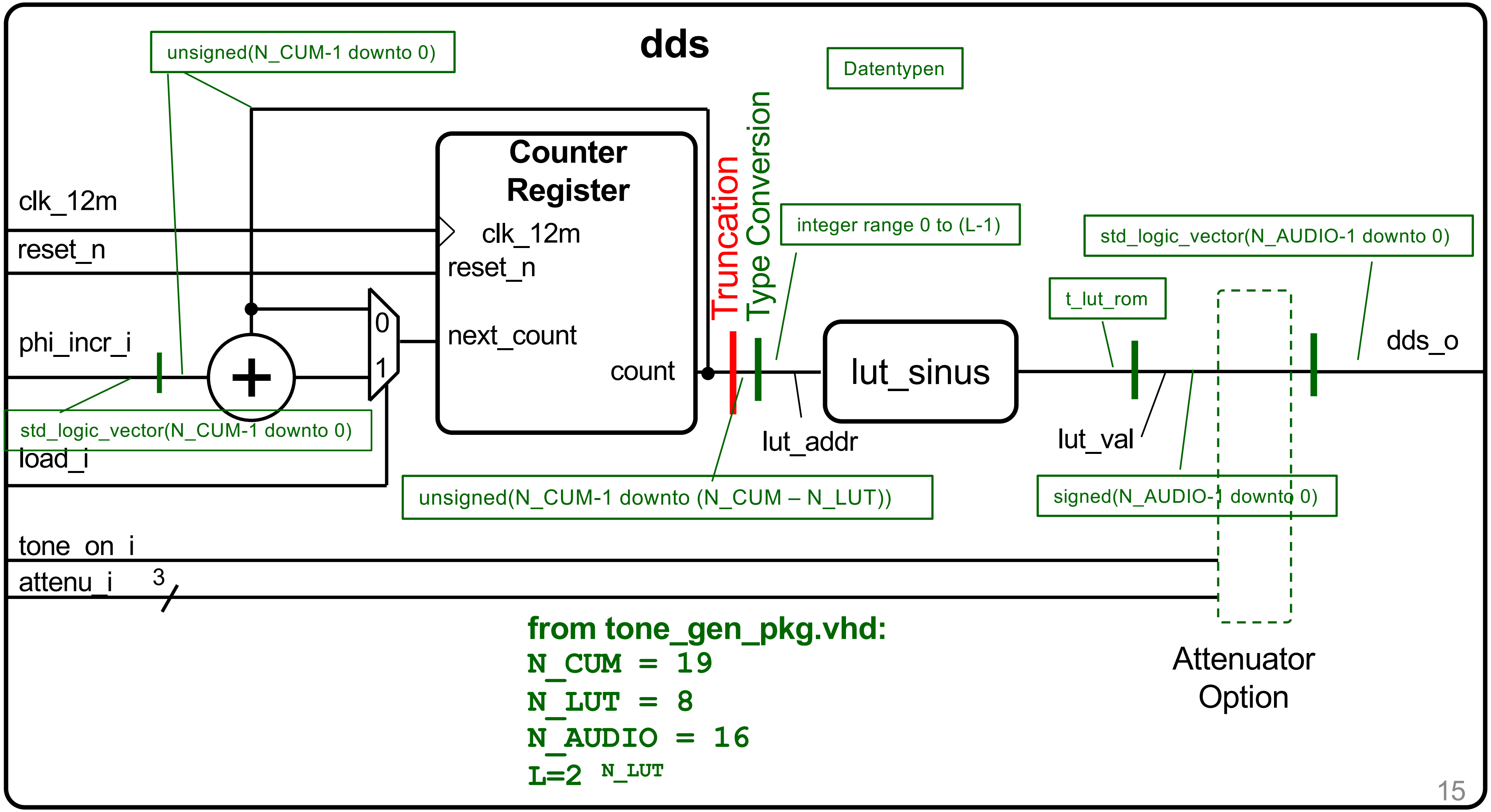
Example:

DO-C4 tone ~261.63Hz

$$\boxed{M = \frac{f_{sig} * L}{f_s} = \frac{261.63Hz * 2^{19}}{48kHz} = 2858}$$

Aufbau DDS Block

School of Engineering



from tone_gen_pkg.vhd:

N_CUM = 19

N_LUT = 8

N_AUDIO = 16

$L=2^{N\_LUT}$

1) Wenn load=1 wird das Counter Register mit dem Wert von next_count aktualisiert

Package tone_gen_pkg.vhd

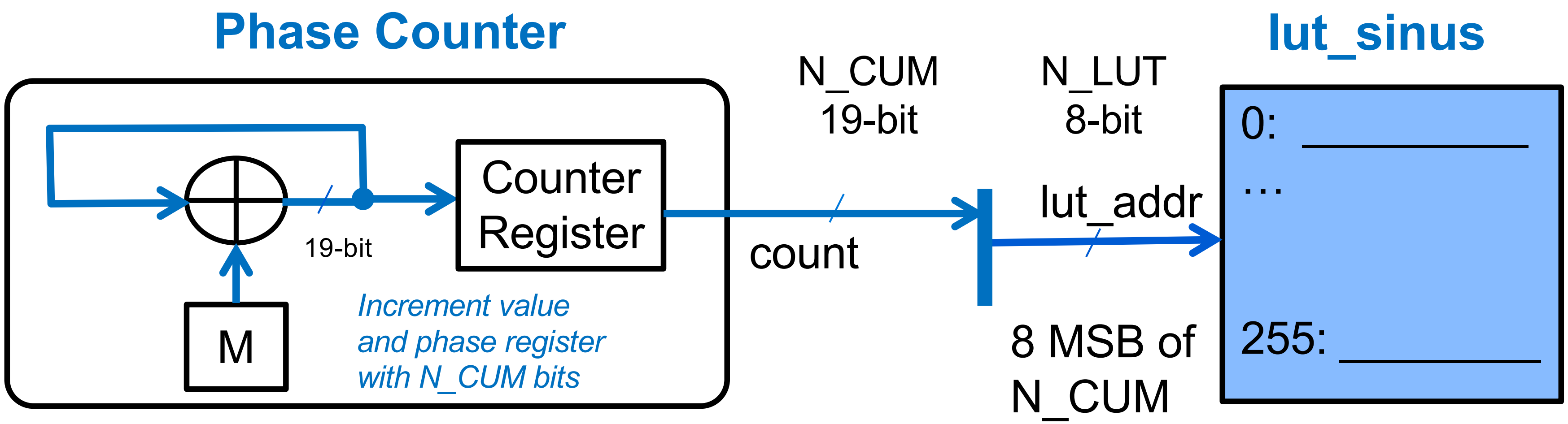
School of Engineering

```
--------------------------------------------------------------------------------
 -- Constant Declaration
--------------------------------------------------------------------------------

   constant N_CUM:   natural :=19;          -- number of bits in phase counter
   constant N_LUT:   natural :=8;           -- number of bits in LUT address
   constant  L:      natural := 2**N_LUT;   -- length of LUT
   constant N_RESOL: natural := 13;          -- Attention:1 bit reserved for sign
--------------------------------------------------------------------------------
-- Type Declaration
--------------------------------------------------------------------------------
   subtype t_audio_range is integer range -(2**(N_RESOL-1))to(2**(N_RESOL-1))-1;
                                            -- range : [-2^12; +(2^12)-1]

   type t_lut_rom is array (0 to L-1) of t_audio_range;

   constant lut_sinus : t_lut_rom :=(
    0, 101, 201, 301, 401, 501, 601,7 00, 799, 897, 995, …
    …501, 401, 301, 201, 101, 0, -101, -201, -301, -401, -501, -601, -700, …);
```

Truncation of LUT Address

School of Engineering

```
CONSTANT N_CUM:          natural :=19; --width counter register
CONSTANT N_LUT:          natural :=8; --width counter register
CONSTANT L:              natural := 2**N_LUT; -- length of LUT
…
SIGNAL count:            unsigned(N_CUM-1 downto 0);
SIGNAL lut_addr :        integer range 0 to L-1;

lut_addr <= to_integer(count(N_CUM-1 downto N_CUM - N_LUT));
```



Audio Output (accessing lut_sinus)

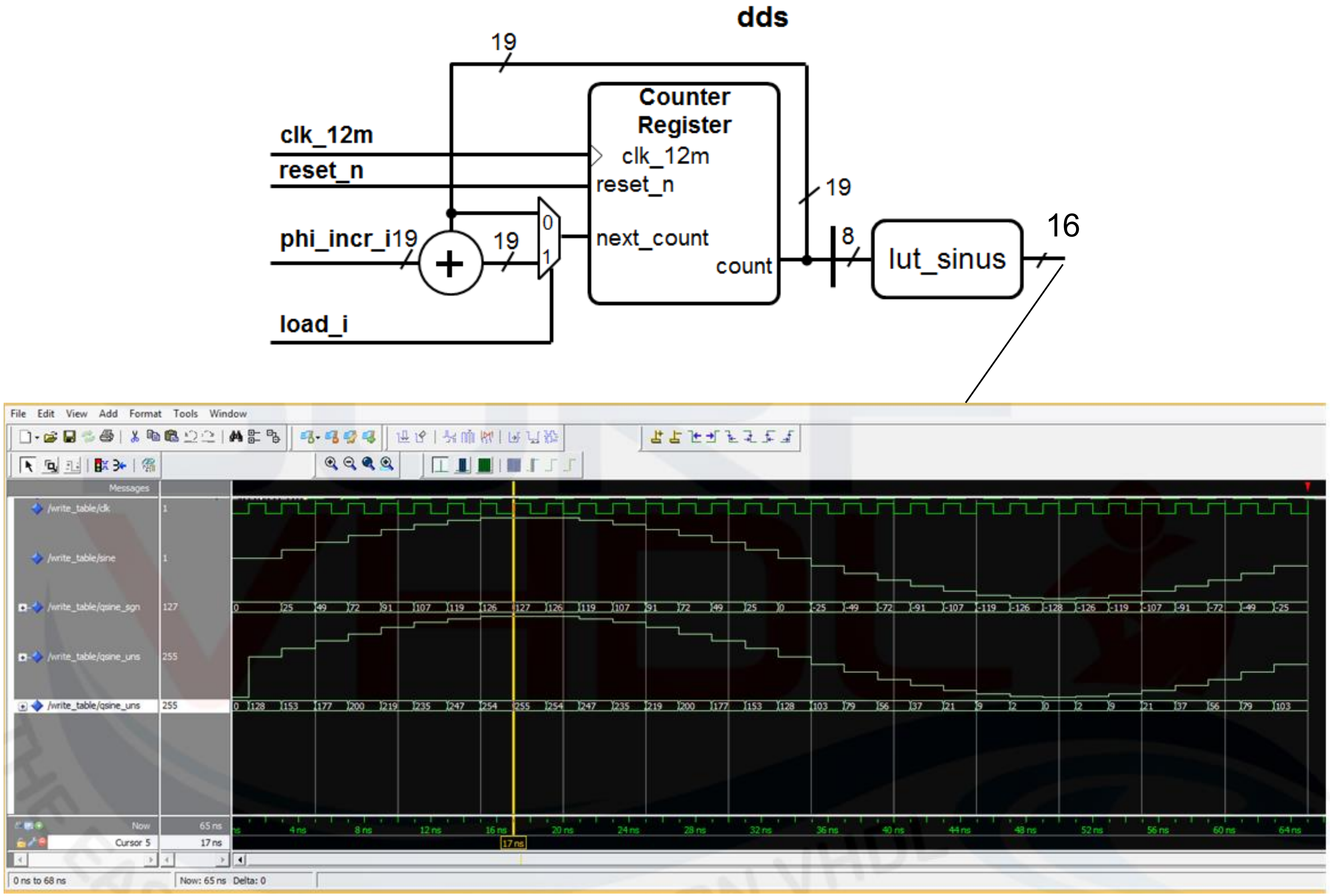
School of Engineering

- Syntax to grab N_LUT MSBs and use as address to search value in LUT
  remember to convert address to integer before using it as index!

```
CONSTANT N_CUM:          natural :=19; --width counter register
CONSTANT N_LUT:          natural :=8; --width counter register
CONSTANT N_AUDIO:        natural :=16; --parallel audio bus width
CONSTANT L:              natural := 2**N_LUT; -- length of LUT
…
SIGNAL lut_val  :        signed(N_AUDIO-1 downto 0);
SIGNAL lut_addr :        integer range 0 to L-1;


…
lut_val <= to_signed(lut_sinus(lut_addr), N_AUDIO);
```

Simulation of LUT Output

School of Engineering



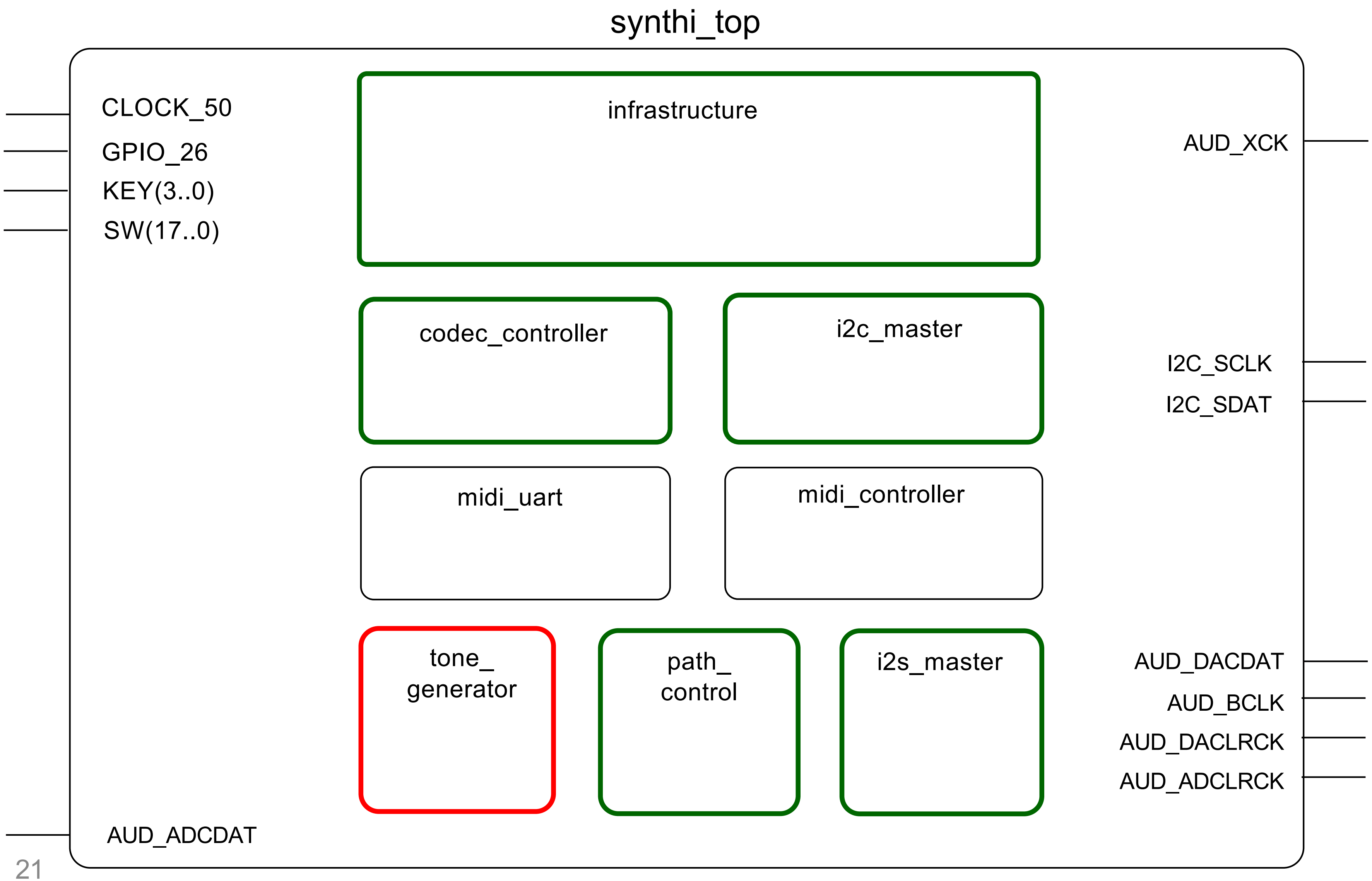
Attenuator

School of Engineering

```
atte := to_integer(unsigned(attenu_i));

case atte is
when 0    => dds_o <= to_integer(lut_val);
when 1    => dds_o <= to_integer(shift_right(lut_val,1));
when 2    => dds_o <= to_integer(shift_right(lut_val,2))
…
when others => …
end case;
```

Synthi Top MS3

School of Engineering



MS3: Tone Generator

School of Engineering

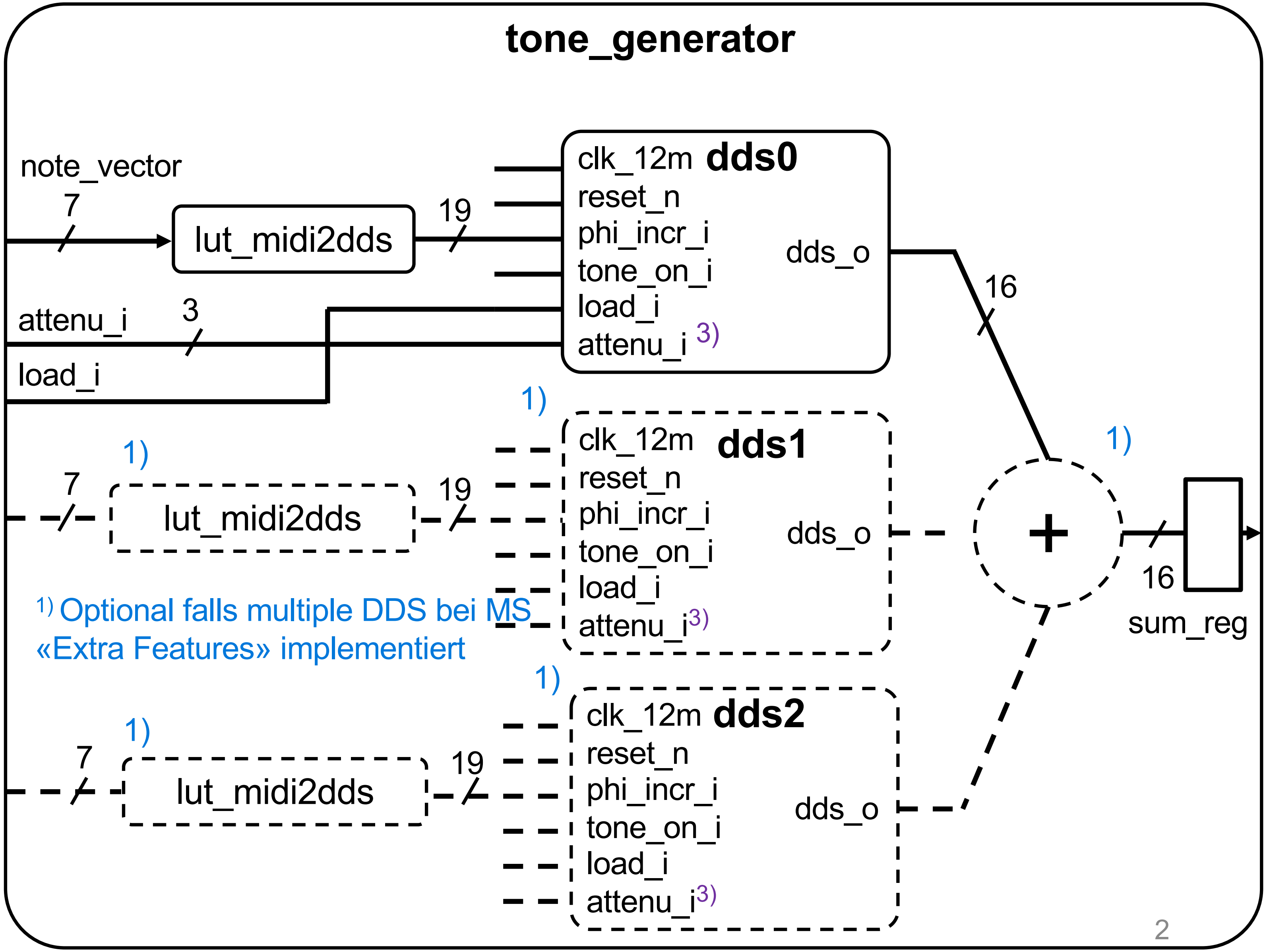
sw_sync(10..4)[2)]

sw_sync(16..17)[2)]

sw_sync(15)[2)]

[2)] Nur für Simulation in synthi_top mit sw_sync verbinden. In Testbench SW mit gpi_sim verbinden. Signale werden nach Implementierung des midi_controller ersetzt

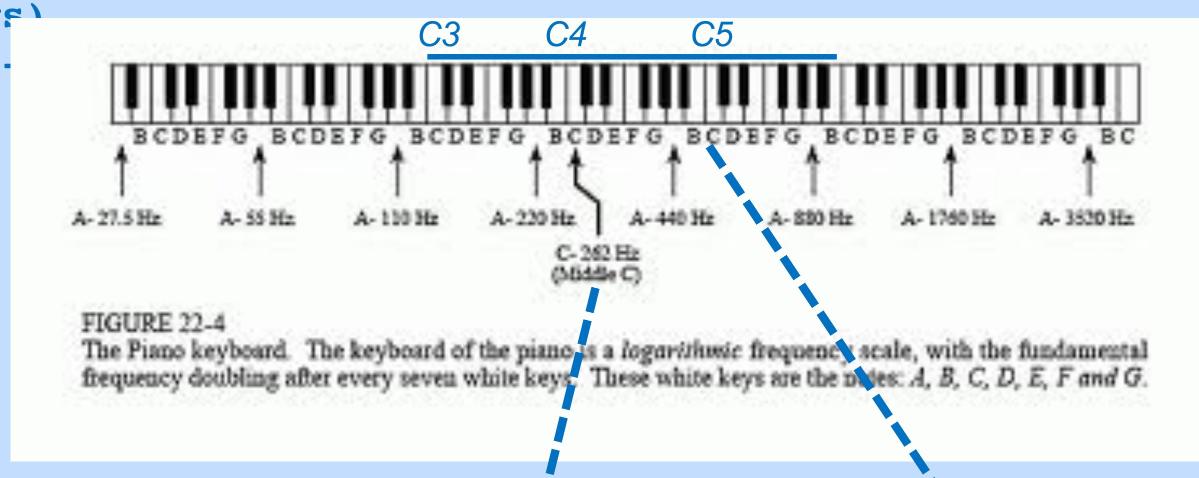
[3)] Optional falls Volume Control gewünscht

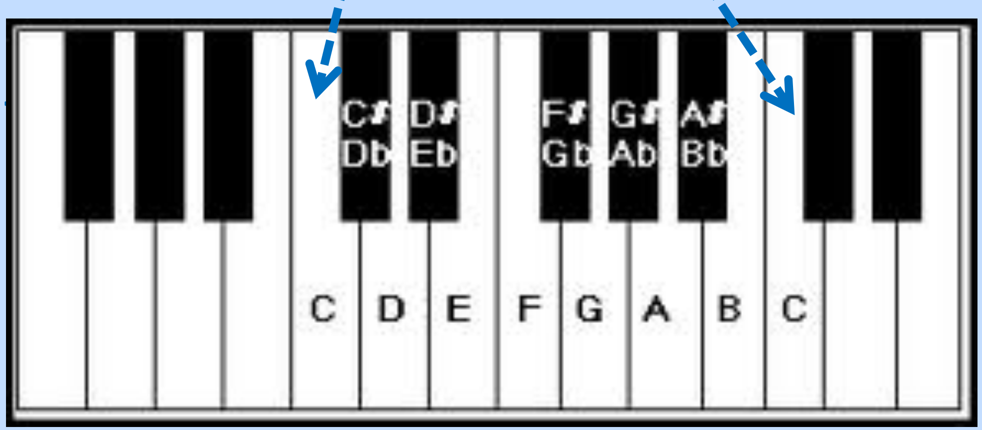


Aufbau lut_midi2dds

- Increment values are pre-calculated and coded as constants in **tone_gen_pkg.vhd**

```
-- Piano Mid-Octave (white keys)
--------------------------------
-- DO-C4   tone ~261.63Hz
-- RE_D4   tone ~293.66Hz
-- MI_E4   tone ~329.63Hz
-- FA_F4   tone ~349.23Hz
-- SOL_G4  tone ~392.00Hz
-- LA_A4   tone ~440.00Hz
-- SI_B4   tone ~493.88Hz
-- DO_C5   tone ~523.25Hz

-- Piano Mid-Octave (black keys)
--------------------------------------------
-- DOS_C4S   tone ~277.18Hz
-- RES_D4S   tone ~311.13Hz
-- FAS_F4S   tone ~369.99Hz
-- SOLS_G4S  tone ~415.30Hz
-- LAS_A4S   tone ~466.16Hz
```





Example:

```
constant M_LA_A4: unsigned(N_CUM-1 downto 0):= to_unsigned(4806, N_CUM));
-- LA_A4   tone ~440.00Hz
```

$M = 2^{19} * 440/48000$

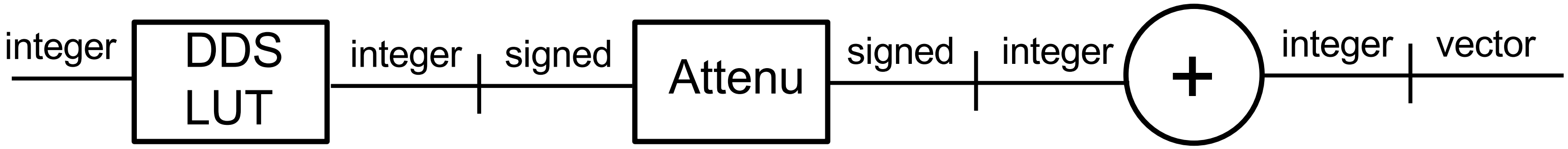
LUT_midi2dds

School of Engineering

```
type t_lut_note_number is array (0 to 127) of
                                std_logic_vector(N_CUM-1 downto 0);

    constant LUT_midi2dds : t_lut_note_number :=(
           CM2_DO  ,   -- Key 0
           CM2S_DOS,   -- Key 1
           DM2_RE,     -- Key 2
           DM2S_RES,   -- Key 3
           EM2_MI,     -- Key 4
           FM2_FA,     -- Key 5
           FM2S_FAS    -- Key 6
           …
           E8_MI,      -- Key 124
           F8_FA,      -- Key 125
           F8S_FAS,    -- Key 126
           G8_SOL      -- Key 127
           );
```

Umwandlungen im Signalverlauf

School of
Engineering



13 bit = -4096 bis +4095 = 8‘192 → 16-bit = 65‘536

Add-In DDS Outputs for Polyphony

School of Engineering

```
comb_sum_output : process(all)
  variable var_sum : integer range -(2**(N_AUDIO-1)) to (2**(N_AUDIO-1))-1;
begin
  var_sum := 0;
  if strobe_i = '1' then
   dds_sum_loop : for i in 0 to 9 loop
     var_sum := var_sum + dds_o(i);
   end loop dds_sum_loop;
   next_sum_reg <= var_sum;
  else
   next_sum_reg <= sum_reg;
  end if;
end process comb_sum_output;

reg_sum_output : process(clk, rst_n)
begin
  if rst_n = '0' then
    sum_reg <= 0;
  elsif rising_edge(clk) then
    sum_reg <= next_sum_reg;
  end if;
end process reg_sum_output;
```