Onderwerp:

Datum:

PROJECT

- We can chose wether to select from Raspery or DE1 SoC
- The application is controlled by a user interface
- One of the sensors has to be used
- Real time behaviour.

5.1

- Pressing two keys simultaneously NOT SUPPORTED
- 12 (Full / half) tones A, Ais, B, C, Cis, D, Dis, E, F, Fis, G, Gis

  A_+, Ais_+, B_+, C_+, Cis_+, D_+, Dis_+, E_+, F_+, Fis_+ ...

- The tone A has freqeuncy 440 Hz

  A^+ has freqeuency 880 Hz

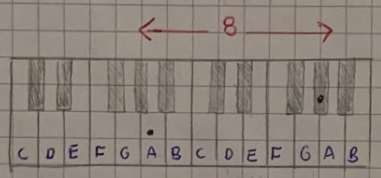
  There is a constant → constant C

  A, Ais

  $tone_{i+1} = C \times tone_i$ and $tone_{i+12} = 2 \times tone_i$

  $C^{12} = 2$

  $C = 2^{\frac{1}{12}} \sim 1,05946$

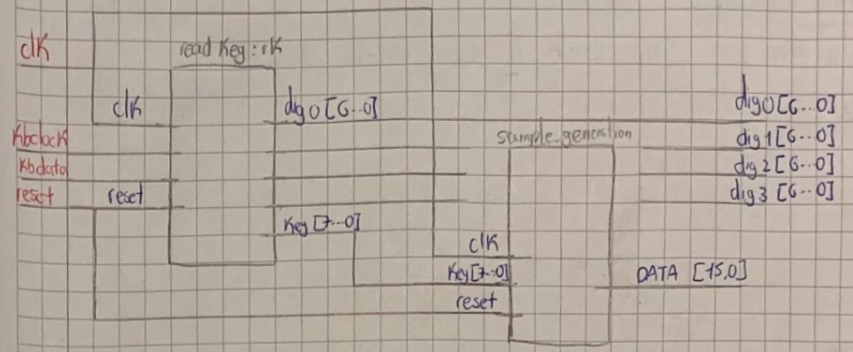


- Every time key "A" the octave is incremented with one, until highes octave.
- Every time key "Z" all tones are decreased one octave until the lowest is reached
- If press in A when the highest octave is reached, no effect
- 6 octaves in this organ, we can shift six time

Onderwerp:

Datum:

5.1.2 Recognize and Generate a code



- Kbclock and Kbdata are used with the Keyboard
- The keyboard generates a code accordin with the pressed key

e.g. 00_{16} TAB

When holding down: the code is repeatedly generated

When released: two consecutive bytes are realised

$F0_{16}$ and the one of the key 00_{16} fist and then

Read Key

- Read one byte. output is a constantly present of the corresponding Key code

Sample Generation

Not equal to 00_{16} is going to generate a sinusoide

↳ no sound if 00_{16}

5.1.3 Read Key

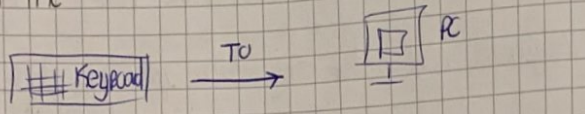
There is 101 Keys 8 bits make $2^8 = 256$ different combinations.

When an extended key is pressed E0 is sent fist, then F0 and then the pressed key

Onderwerp:
Datum:

5.1.3 ReadKey

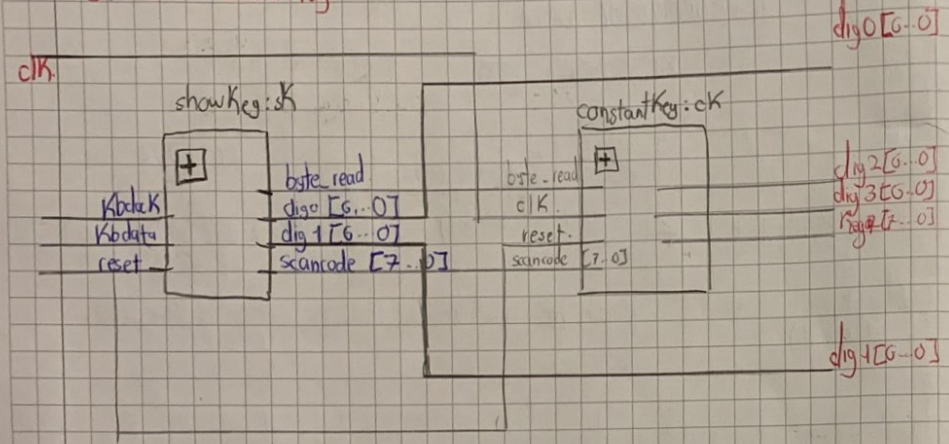
Interested from the



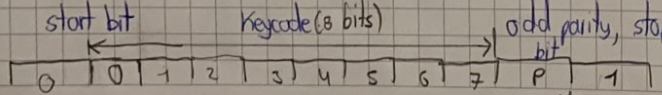
1. When a key is pressed a byte is generated <Keycode>
2. Manteniendo presionado <Keycode>, <Keycode>, <Keycode>
3. When released.

$F0_{16}$, <Keycode of the released key>

5.1.3.1 ShowKey

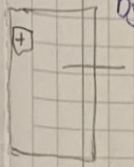


get its input from KBCLOCK and KBDATA.
it has an asynchronous reset.
When KEY0 is pressed the reset is applied.
When { NOTHING is pressed KBCLOCK is continuously high.
PRESSING a key, 11 bits are sent.



Onderwerp:
Datum:

1. PARITY BIT IS IGNORED.
2. KBDATA is stable on the FALLING EDGE of KBCLOCK.



byte_read is '1' when the 11 bits transmitted bits are received.

Every time the keyboard send a byte (e.g when holding or releasing the key). byte_read is first time '0' and again '1' after it has received the 11 bits

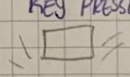
IN Hexadecimal notation

USED for debbuging DIG0 4 least significant bits
DIG1 4 most significant bits

5.1.3.2 ConstanKey

The clock of the board (50MHz)

BYTE_READ indicates SCANCODE is valid.



the corresponding scan code is a constant value available at the output of CONSTANTKEY.

The input SCANCODE is used when in the previous clock cycle (50MHz) the input BYTE_READ was "0" and the current input is "1"

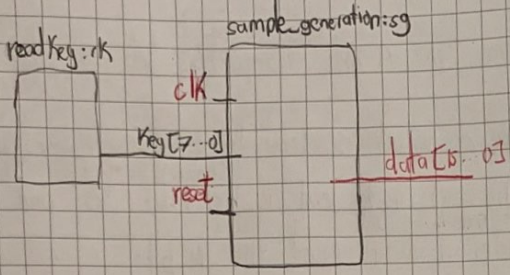
The outputs DIG3 and DIG2 show the constant values in hex of the 4 most significant values and the 4 least significants (used for debbuging)

Onderwerp:

Datum:

5.1.4. Sample Generation

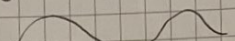
has 4 subsystems



1. SINEGEN.

the output is a discrete sine wave (16 bits, signed number)

it has an LUT to produce the wave



when next_value is '1' the next entry in the LUT is read.

2. KEY2PULSELENGTH

when a Key is pulsed a sine is generated

The f (frequency) depends on the Key pressed

the output PULSE_LENGTH is a constant value that is the number of clock cycles before the next entry from the LUT in SINEGEN is read.

3. MUL_POWER_OF_2

When Key 'A' or 'z' is pressed an octave Change is reliazed.

↳ double frequency

halved frequency

output PULSE_LENGTH_MULTIPLIED constant value for a given tone

Onderwerp:

Datum:

4. PULSELENGTH2NEXTVALUE

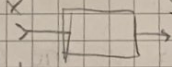
↳ This generates the pulse for SINEGEN.

5.2. VHDL subset: guidelines must be used

5.2.1 SYNCHRONOUS SYSTEM

```
process (reset, clk).
  < here local declarations >
  begin.
  < no VHDL code here >
      if reset = '0' then
      < reset the variables / signals assigned to in this process >.
      else if falling_edge (clk) then.
      < here the behaviour (sequential order)
      end if;
  < no VHDL code here.
end process;
```

5.2.2 COMBINATIONAL LOGIC



the output depends at all times of the input

have no feedback.

- can be modelled with concurrent statements
- When running a PROCESS statement
  - has a sensitivity list (list of signals)
  - All signals read in the process has to be in the sensitivity list
  - A variable must be assigned a value before is read
  - In the process statement don't use 'falling_or

Onderwerp:

Datum:

5.2.2.1 Combinational loop

if during synthesis:

Warning: Timing Analysis is analyzing one or more combinational loops as latches

Latches are not allowed

5.2.3 Difference between variables and signals

- variables are always updated immediately. :=
- signals not <=

An exception:

JOURNAL.

VHDL DESCRIPTIONS

- VHDL test benches

Relevant parts of the waveform

- with explanation

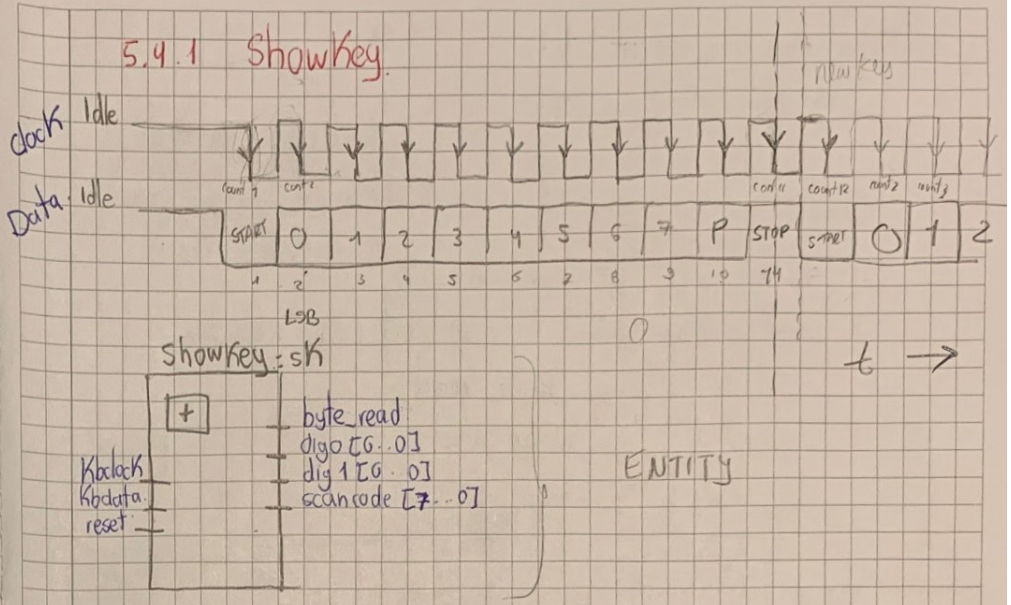
Schematics of the synthesis result

- RLT view

Onderwerp:

Datum:

5.4.1 Showkey



- Your press it KB clock goes "0"
  Unpressed KB clock "1".
- When KBDATA is low and high?
- KEY generates its own clock (20 KHz)
- reset is asynchronous and is alway '1'
- KEY0 reset is applied

KEYBOARD TO DE1-SOC

- Reset is high or low?

Reset makes high off
When we press is '0'
PREVIOUS LAB

Onderwerp:

Datum:

What does the generic do?

is ik where I define variable.

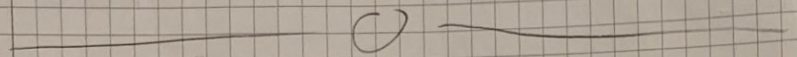
ShowKey

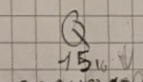
→ show the key pressed

→ send information constant : on subsystem

→ byte_read

→ scancode



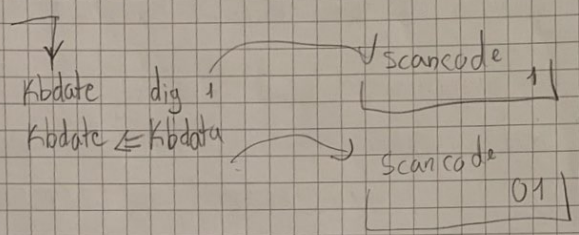


IF reset = '0'

kb clock <= '0'

IFELSE fallin_edge (clk) THEN

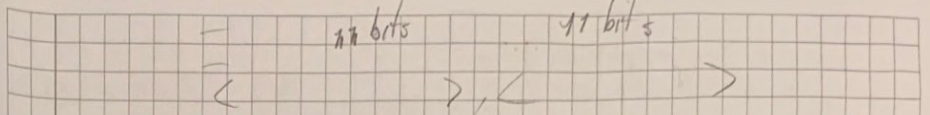
Falling ede



scancode (i)

Onderwerp:

Datum:

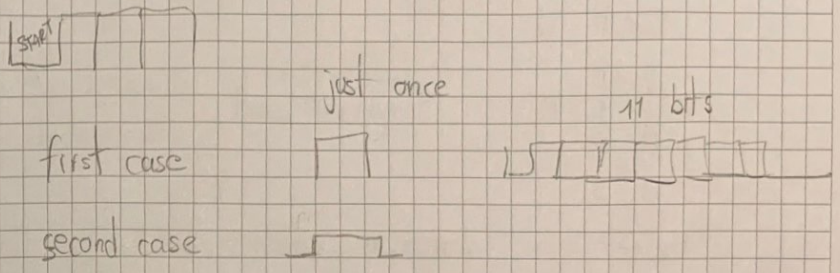


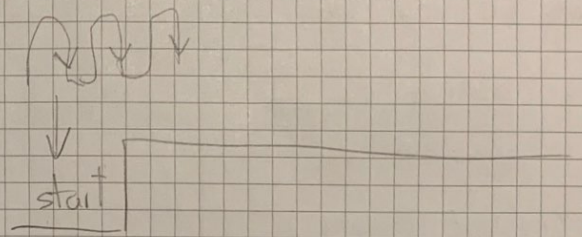
FUNCTION hexzdisplay...

IF reset = '0'

ELSIF falling_edge (Kbclock)

count : count + 1 ;



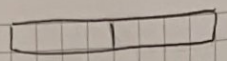


13 bits

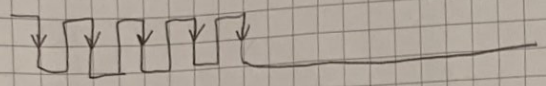
coot = 0

Onderwerp:

Datum:



LSB :
MSB :



count →
if 1
if 2
if 3

scancode (0) = 1

Long (0) = 1

LSB (i) = kadata

count += 1 ;
LSB ⇐ Scancode (3 downto 0)
HSB ⇐ - - (7 4)

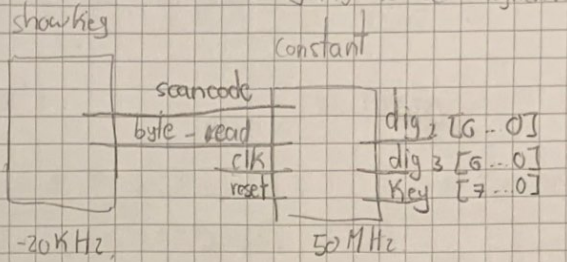
Onderwerp:

Datum:

5.1.3.2 Constant Key

c What do they mean with byte_read changes near the active edge of the clock

d What does represent that byte_read is going to be asynchronous input ?



PROCESS
IF this happens
byte_read Previous TRUE Current THEN
both of them
'0' '1'

NEW_SCANCODE_DETECTED

byte_read go stays longer because clock

PRESENT STATE	NEXT STATE

x: NEW_SCANCODE_DETECTED
y: SCANCODE

Onderwerp:

Datum:

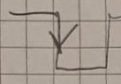
Asynchronus inputs

has a directly effect regardless of the clock

Combinational	Sequentially
• No memory	• Memory
• Output relay in current state	• Previous state and input

We copy in the

Falling edge of the clock



byteread_sync1 ⇐ byte_read ;
byteread_sync2 ⇐ byteread_sync1

elsif rising_edge (clk) then
if prev = '0' and byte_read = '1' then
nsd := '1';
else
nsd := '0';
end if;
prev := byte_read.

Onderwerp:

Datum:

Process (clk, reset) is
Begin
IF
(reset = '0')

byteread_sync1 ⇐ '0';
byteread_sync2 ⇐ '0'
scancode_sync ⇐ "00000000"

current_s
byteread_sync2
scancode_sync

ELSE
falling_edge (clk)

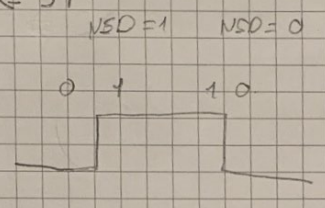
byteread_sync1 ⇐ byte_read
byteread_sync2 ⇐ byteread_sync1

IF byteread_sinc2 = '1'
scancode_sync ⇐ scancode

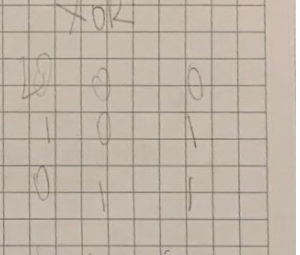
INPUT DECODER Process (current_s, byteread_sync2, scancode_sync)

case current_s

when S0
if (byteread_sync = 1) then
next_s ⇐ S1



clk
if rising_edge(button)

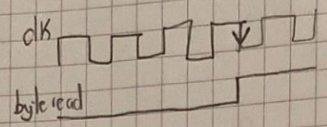


Onderwerp:

Datum:

Proccess

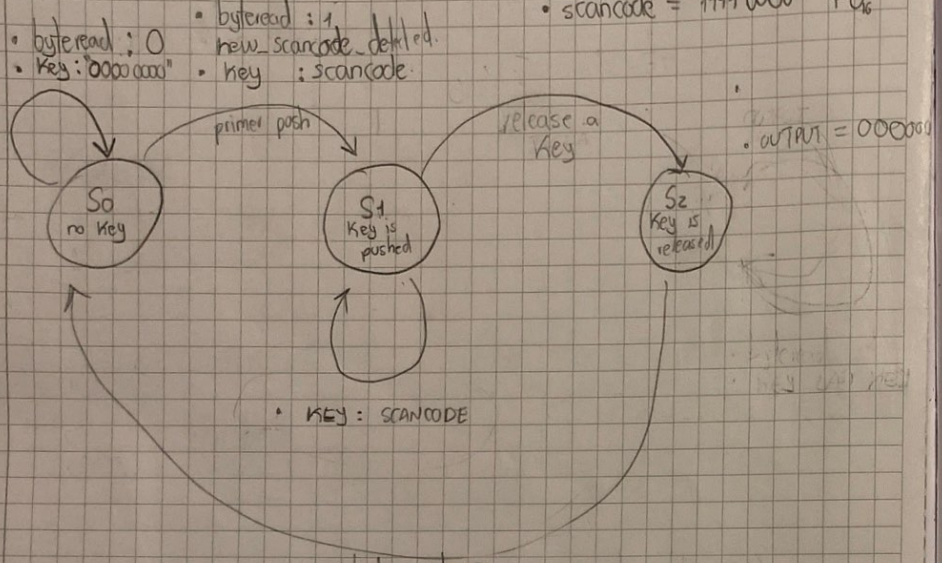
1. In each falling edge of the clk



byteread is checked if it is 1

IF: '1' (NEW SCANCODE IS GENERATED) THIS HAPPENDS

scansync ⇐ scancode

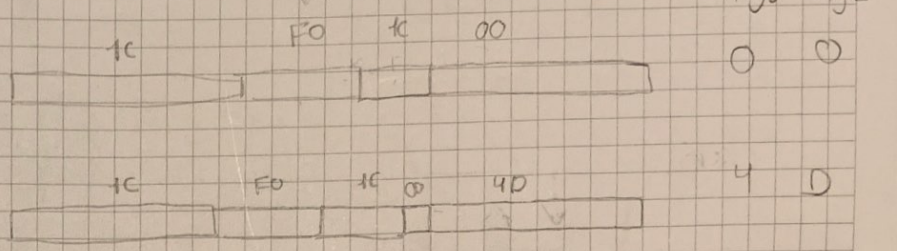


Onderwerp:

Datum:

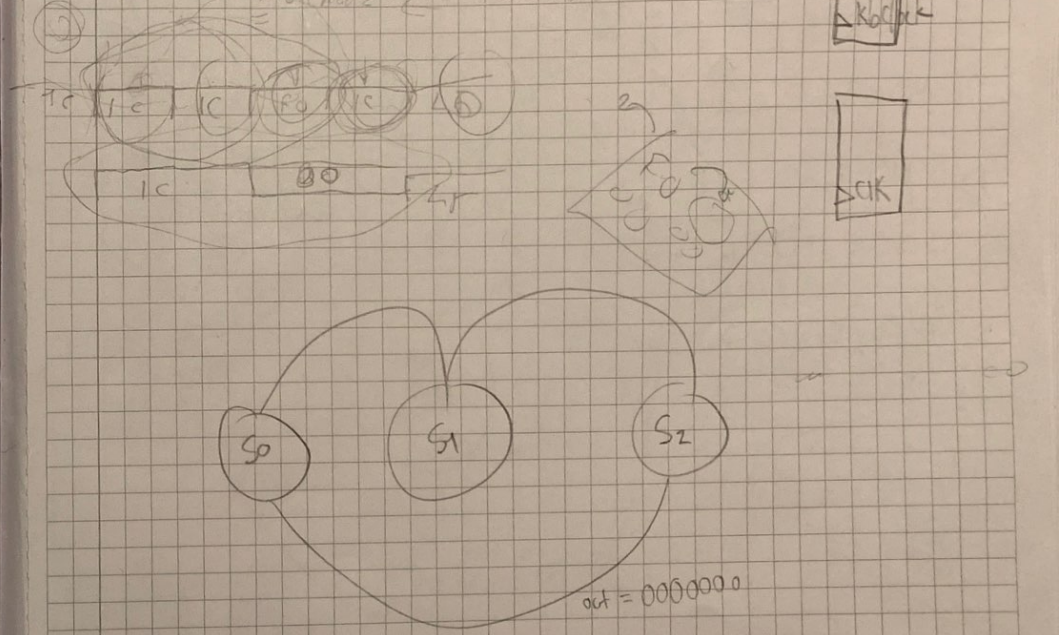
IF Previous_S := 0 AND Current state := 1

NEW_SCANCODE_DETECTED



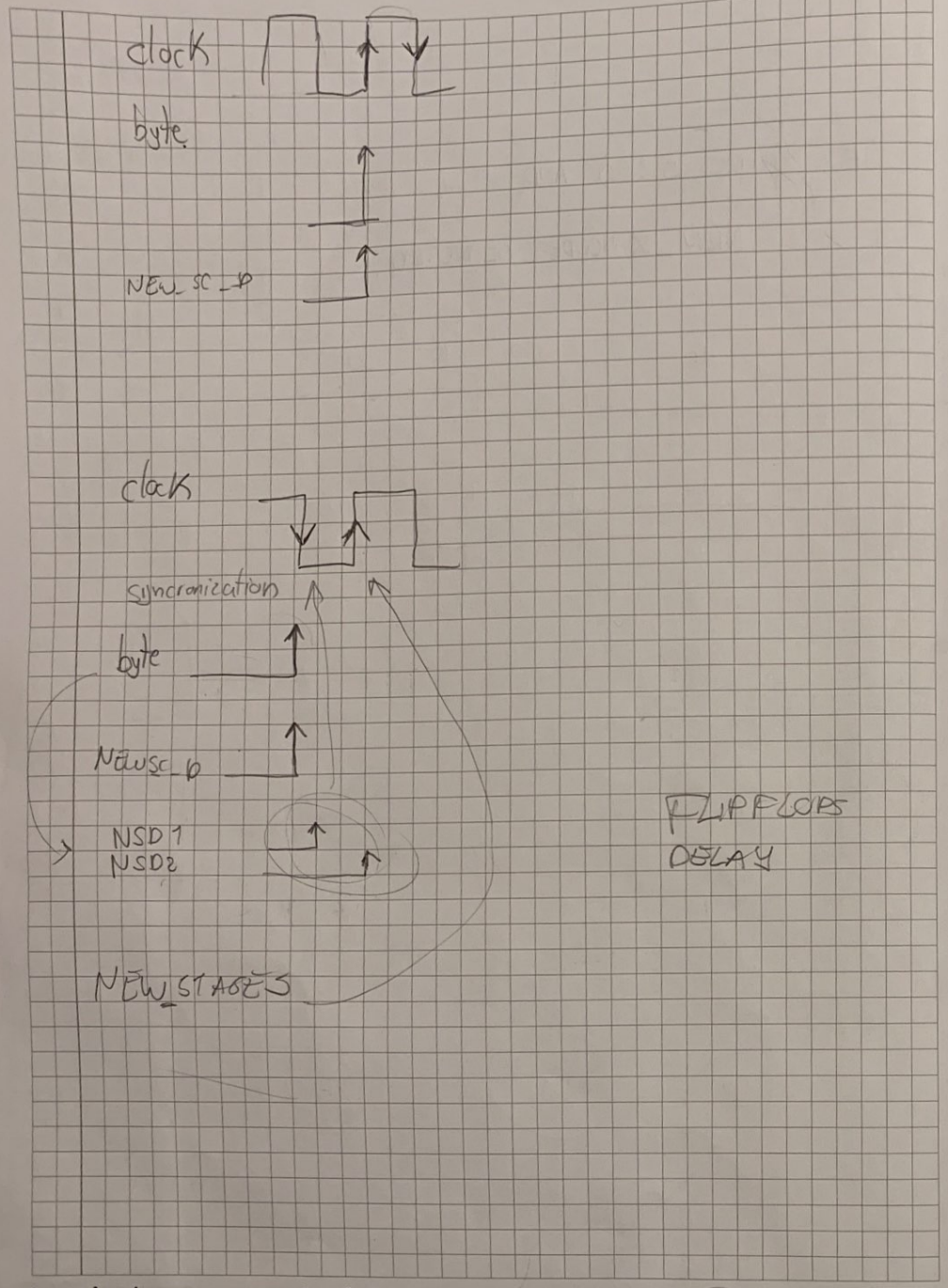
byteread ← byteread

byteread2 ← byteread1



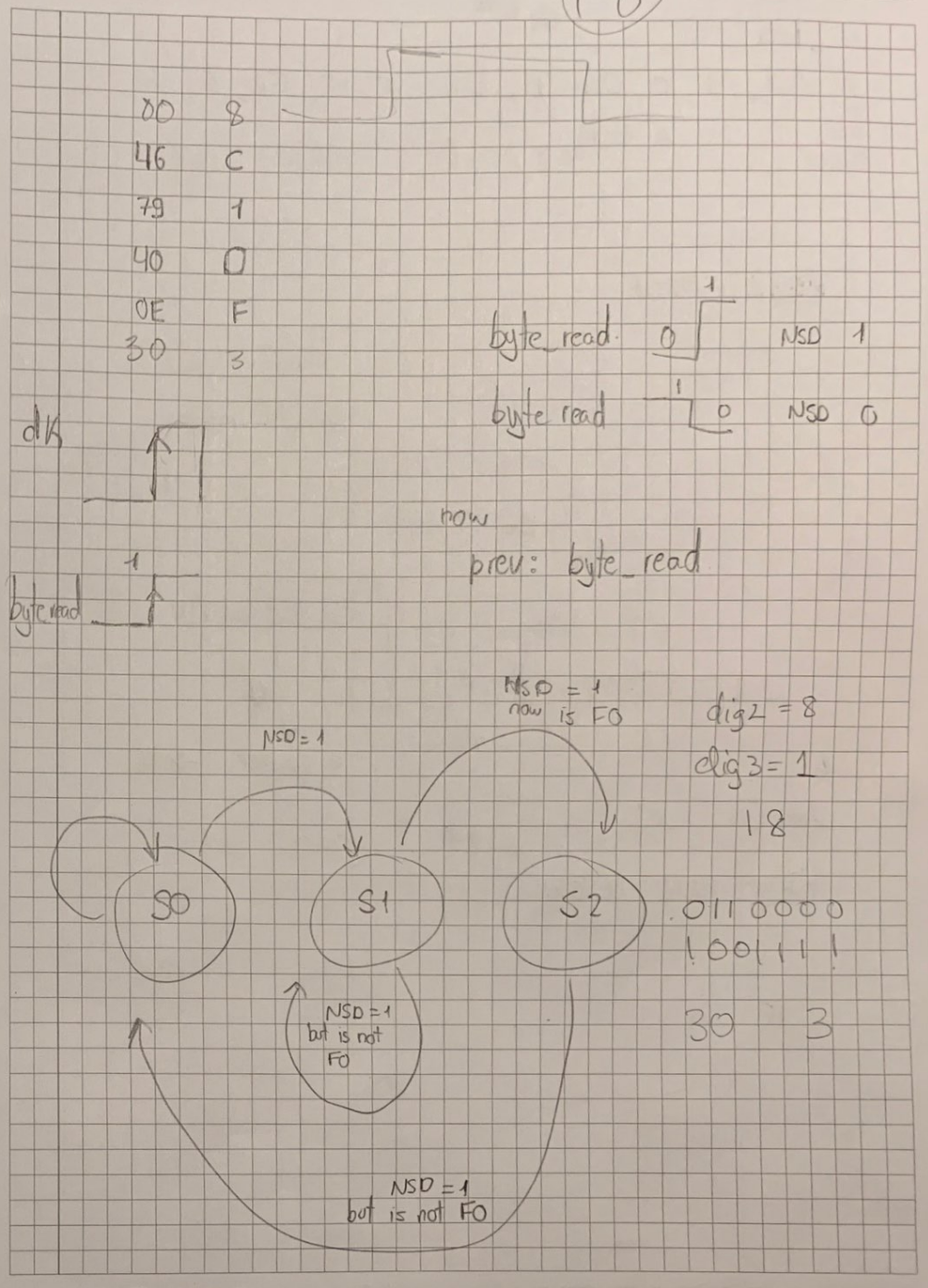
Onderwerp:

Datum:



Onderwerp:

Datum:

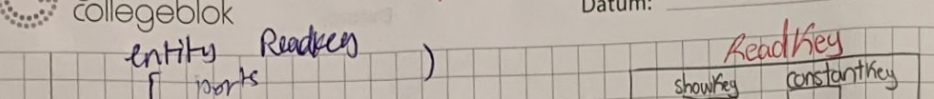


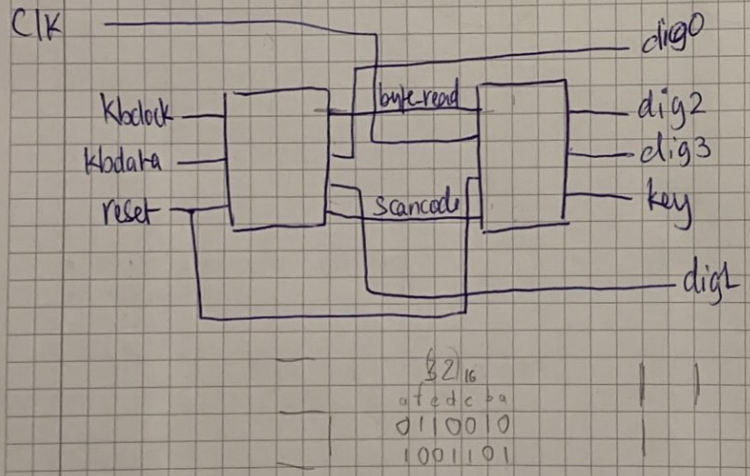
5.4.4 Portmap Readkey

Onderwerp:

Datum:

```
entity Readkey
( ports
)

architecture bhv of readkey is

Component Showkey
    port ( byteread
           scancode
           Kbclock
           Kbdata
           reset ] in

           byte_read
           digo
           dig1
           Scancode ] out
    );
end component;

component Constantkey
    port ( byteread
           clk
           Scancode
           reset ] in

           digz
           dig3
           key ] out )
end component

begin
1: Showkey port map ( kbclock, kbdata..., scancode)
```

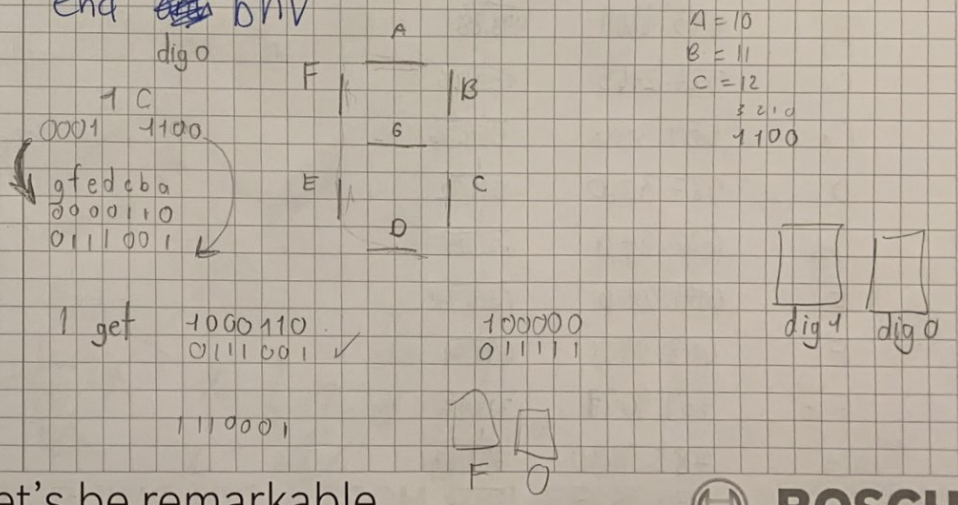




Onderwerp:

Datum:

```
begin
SK: Showkey portmap (kbclock => kbclock,
                     kbdata => kbdata,
                     reset => reset,
                     byte_read => byte_read,
                     dig0 => dig0,
                     dig1 => dig1,
                     Scancode => scancode );

CK: constantkey portmap ( byte_read => byte_read,
                          clk => clk,
                          Scancode => scancode,
                          dig2 => dig2,
                          dig3 => dig3,
                          key => key );

end bhv
```



Onderwerp:

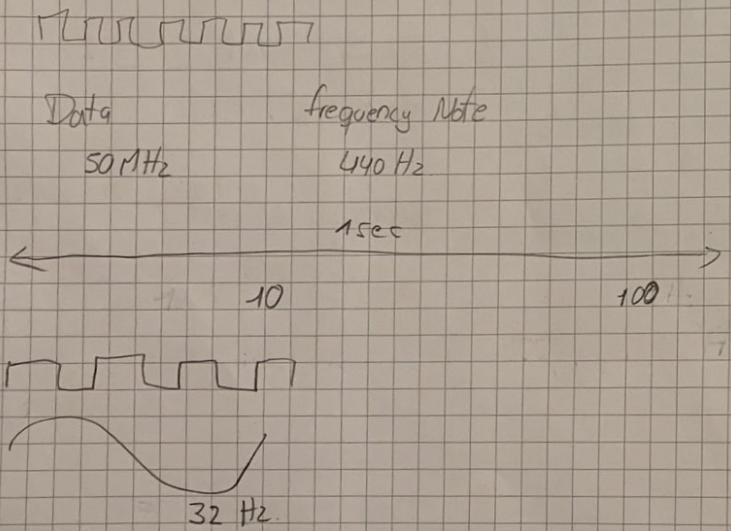
Datum:

5.4.5. Key2pulselength

IF CONSTANT key_TAb: std_logic_vector : x"0D"
Then: Pulse length: 3203

IF CONSTANT KEY_Q: std_logic_vector: x"15"
then Pulse length: 7412

When others:
Pulse length: 0



Onderwerp:

Datum:

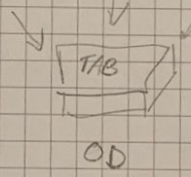
delete

8 Notes

A = 440
B = 493,88
C = 523,25
D = 587,33
E = 659,25
F = 698,46
G = 783,99
A+ = 880

Input

KEY: (7 DOWN TO 0)



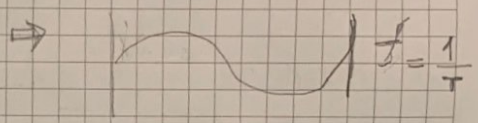
KEY (0001110)

1C
00011100

Output

PULSELENGTH: Integer

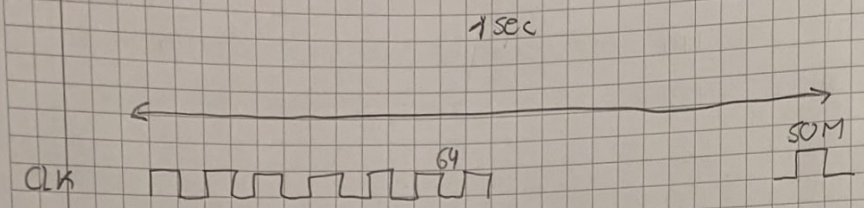
Determinada Frequency

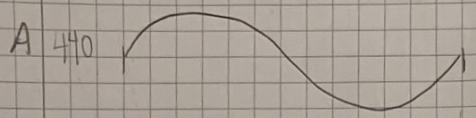


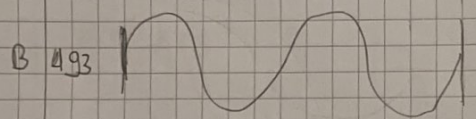
PULSE_LENGTH: { Integer 0 - 20 000 }

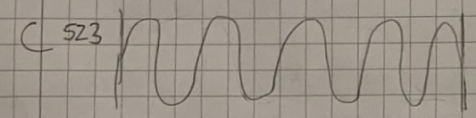
Onderwerp:

Datum:

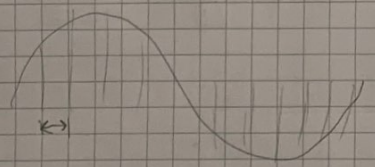








- Length of each pulse in clock cycles



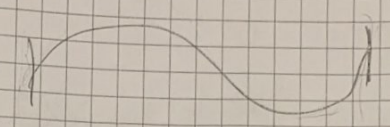
each step has a certain length
each length

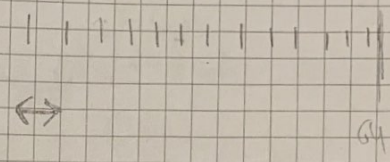
Onderwerp:

Datum:

440 Hz

Period: 0,0022 sec





0,000034

50M cycles → 1 sec

x → 0,000034

$$x = \frac{50M \cdot 0,000034}{1}$$

1775

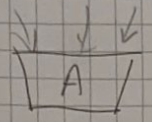
$$x = 50M \cdot \frac{1}{440} \cdot \frac{1}{64}$$

To calculate the integer value.

$$X = \frac{50 \times 10^6}{64 \times freq}$$

When

Mul_power_of_2



freq is double

pulse_length x2

if 55 ≤ pulse_length ≤ 1480 and key = x"1C"

pulse_length_mult ← pulse_length x 2

out of the limit

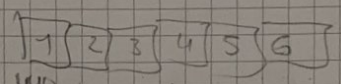
27, 5 28. 409.

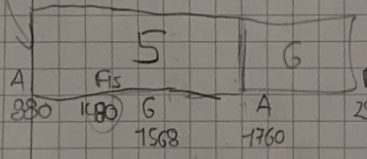
50 M
440 x 64

Octaves

1	2	3	4	5	6
55	110	220	440	880	1760

Does it has to start A in





Fist

A G

if (55 ≤ pulse_length ≤ 1480) and key = x"1C"

pulse_length.

* Can I add nomeric_bit?
* Igonna round to

~~X := '0' & X(- downto 1)~~ multiply

X := X(- downto 0) & '0'

(5) 101 & '0'

number

1010

10

23 10111 101110

46

pulse_length(15 downto 1)

'0' & 1010

0101

shif one position to the ←

440 $A_3 A_2 A_1 A_0$

00000 1101 1000

1101 1100

gratis collegeblok 5.4.7 PulseLength2NextValue Onderwerp:
PulseLengthNextValue Datum:

Generates a pulse for Sinegen

Input

Sine_complete (std_logic)

pulse_length (integer) 0-20000

Output

Next value (std_logic)

'1' for 1 clock cycle when new pulse length is inputted

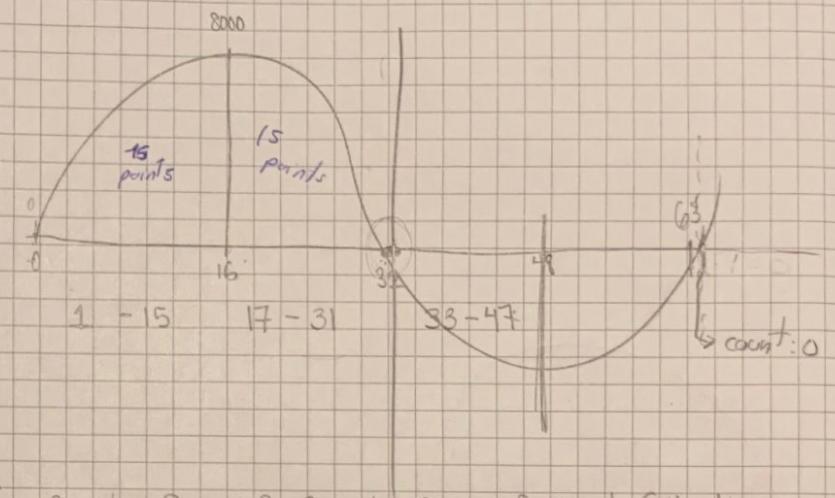
'0' when pulse length is 0

```
Var count integer 0 to max_length := 0
Var length : integer

IF reset='0'
Next_Value = 0
count := 0      length := 0
Elsif falling_edge(clk)

If pulse_length = 0
Next_value <= 0
End if

If sine_complete = '1'
   length := pulse_length;
   count := 0
Endif

If count = length then
   count := count + 1;
Else

If pulse_length = 0 then
   next_value <= 0;
Else
   If count < length then
      count := count + 1;
   Elsif count = length then
      Next_value <= '1';
   Endif
End If
```

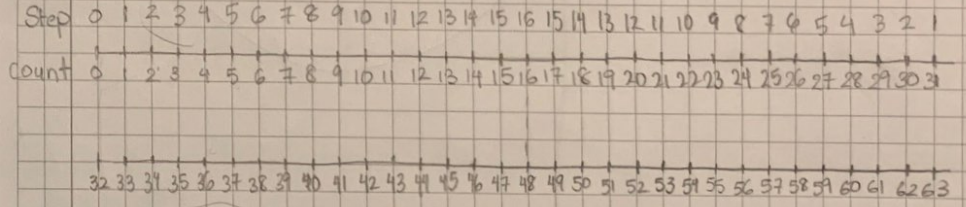

gratis collegeblok 5.4.8 Sine gen Onderwerp:
Datum:

A x Sin (2π x/64) x; steep.

213
-198
-7
15



IF count = 0 OR count = 32 OR count = 64 then
DATA <= 0000
END IF



If (count = 0 - 16) : DATA <= std_logic_vector (to_sign(LUT(count)))

If (count = 17 - 31) : DATA <= std_logic_vector (to_Sign(LUT(32 - count)))

If (count = 32 - 48) : " " " " " " LUT(count - 32)

If (count = 49 - 63) : " " " " " " LUT(64 - count)

Onderwerp:

Datum:

N=15 (bits)

$2^{15}-1 = 32767$

1 2 3. 4th octave 5 6

↓

A

27,5 55 110 220 440 880

↳ 28 409 integer value is not in the range 0 – 20000

type key IS (Tab, 1, Q, W, 3, ..., 1)

Case key IS

when Tab => Pulse_length := ;

when 1 => Pulse_length := ;

:

when 1 => Pulse_length <= ;

END CASE;

Sinegen

Onderwerp:

Datum:

$2^{16} = 65536$

$2^{13} = 8192$

Output

DATA: 16 bit signed number

↳ a discrete sign wave

Sine_complete → '1' when sign is completed/otherwise 0

Input

Next_value: '1' or '0'

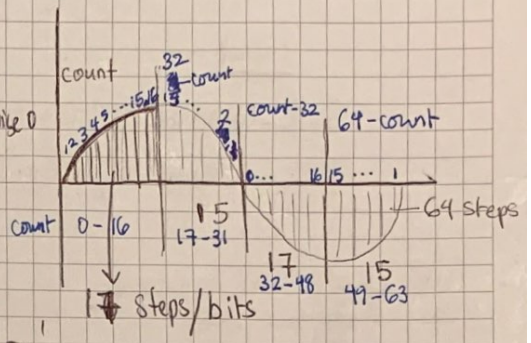
↳ when '1', next value in LUT is read

~~↳ 1 when sine is completed/otherwise 0~~

reset

When ~~sine comp~~ is applied, Sine_complete is '1'. why?

LUT ~~descri~~ approximates sine wave



Amplitude: ~~(8000)~~ $_{10}$

8000

$A \sin\left(\frac{2\pi x}{64}\right)$

A. 25 Determine content of LUT.

A. 26 Give VHDL description

A. 27 Verify Sinegen

A. 28 Synthesise wit Quartus

A0 Step(x)	Value	
1	784.1	784
2	1560.7	1561
3	2322.3	2322
4	3061.5	3062
5	3771.2	3771
6	4444.6	4445
7	5075.1	5075
8	5656.9	5657
9	6184.1	6184
10	6651.8	6652
11	7055.4	7055
12	7391.0	7391
13	7655.5	7656
14	7846.3	7846
15	7961	7961
16	8000	8000

Function LUT

takes: step

returns: value

~~step~~ case step

when 0

1

2

16

~~return~~ ~~step~~ return step

architecture:

If reset '0' then

Sine_complete <= '1'

Elsif falling_edge(clk) then

If Next_value = '1' then

If ~~LUT~~ count

DATA <= LUT(0

Onderwerp:

Datum:

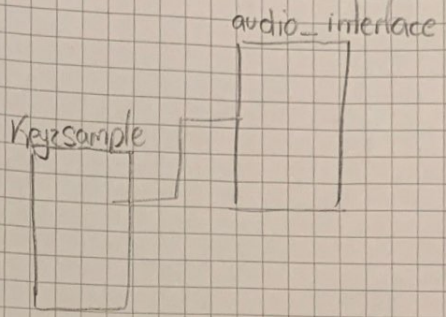
5.4.9 Sample Generation

```
Entity sample_generation IS
    ;

Architecture behaviour of sample_generation is
  Component key2pulselength
  Generic (max_length: integer := 2000);
PORT ( KEY: IN
       pulse_length: OUT )

Component mul_power_of_2
    Port ( clk
           ; )

Component pulselength2nextvalue
    Port (clk: ~
          ; )

  Component Sinegen
    Port ( ... )

  Signal pulse_length
  Signal pulse_length_multiplied
  Signal next value
  Signal sign complete

Begin
  K2PL : key2pulselength Portmap
                          ( ; )

  MULPOF2 : mul_power_of_2 Portmap
                          ( ; )

  PL2NV : pulselength2nextvalue portmap
                          ( ; )

  SG: Sinegen portmap ( ; )
```


Onderwerp:

Datum:

5.4.10 Key2Sample




```
Component readKey is
  ports ::
end Component;
Component Sample_generation is
ports ::
end component.

Begin
  rk: readKey port map (   clk => clk,
                           Kbclock => Kbclock,
                           Kbdata => Kbdata,
                           reset => reset,
                           dig0 => dig0,
                           dig1 => dig1,
                           dig2 => dig2,
                           dig3 => dig3,
                           key => key );

  sd: sample_generation port map (  clk => clk,
                                    key => key,
                                    reset => reset,
                                    DATA => DATA);
```