Goertzel Filter

With VDHL Implementation

- Ali Beiti Aydenlou
- Roghieh Farajialamooti
- Ghazaleh Hadian Ghahfarokhi
- Seda Sensoy

Microelectronics & HW/SW-Co-Design
Summer 2024 Homework

Prof. Dr. Peter Schulz

Fachhochschule
Dortmund
University of Applied Sciences and Arts

Scope

- Definition and Benefits of the Goertzel Algorithm
- Design of a Goertzel Filter according to given specifications
- Using Matlab to generate stimuli data
- Goertzel Filter VHDL implementation
- Development of a test bench
- Test cases
- EPWave Simulation Design
- Verification
- Conclusion

Fachhochschule Dortmund
University of Applied Sciences and Arts

What is Goertzel Algorithm?

- A digital signal processing algorithm
- Efficient computation of individual discrete Fourier transform (DFT) bins
- Detecting the presence of specific frequencies in a signal
- An efficient alternative to the fast Fourier transform (FFT)

Benefits of Goertzel Algorithm?

- Efficiency in Frequency Detection
- Simplicity and Ease of Implementation
- Real-time Processing Capabilities
- Versatility in Applications
- Flexibility in Implementation

Fachhochschule Dortmund
University of Applied Sciences and Arts

MATLAB - Discrete Signal Properties

These parameters are essential inputs for setting up and using the Goertzel Filter.

```
fs = 4e6; % Sampling frequency
f0 = 150e3; % Desired frequency
N = 135; % Number of samples
t = (0:N-1)*(1/fs); % timesteps for the desired number of samples
```

Goertzel Filter Basics

$\alpha = \frac{2\pi k}{N}$

$\beta = \frac{2\pi k(N-1)}{N}$

$a = \cos(\beta)$

$b = -\sin(\beta)$

$c = \sin(\alpha)\sin(\beta) - \cos(\alpha)\cos(\beta)$

$d = \sin(2\pi k)$

$\omega_0 = \omega_1 = \omega_2 = 0$

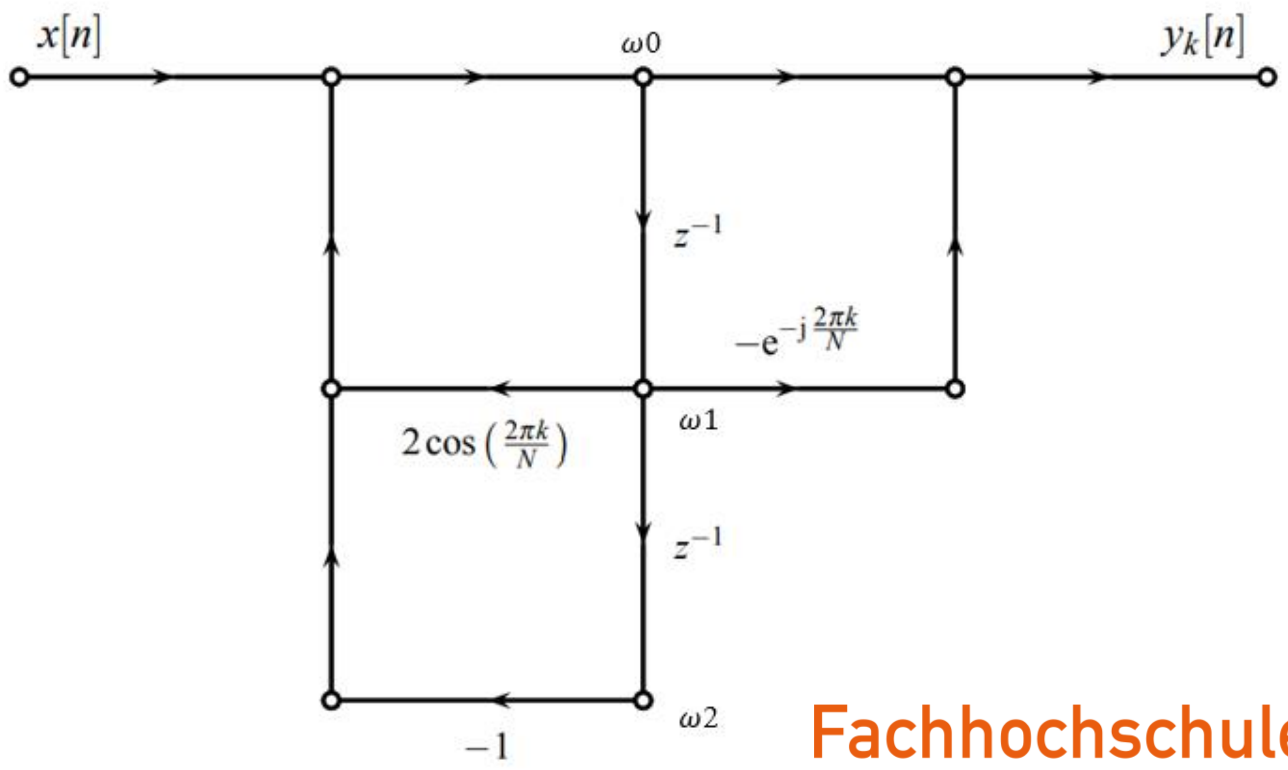
for $window_{index} = 0$ *to* $window_{size} - 1$ **do**

- $\omega_0 = x_{in} + 2\cos(\alpha)\omega_1 - \omega_2$
- $\omega_2 = \omega_1$
- $\omega_1 = \omega_0$

$X_k = a\omega_1 + c\omega_2 + j(b\omega_1 + d\omega_2)$

$|X_k| = \sqrt{(a\omega_1 + c\omega_2)^2 + (b\omega_1 + d\omega_2)^2}$

$\Phi(X_k) = \arctan(\frac{b\omega_1 + d\omega_2}{a\omega_1 + c\omega_2})$



Fachhochschule Dortmund
University of Applied Sciences and Arts

Goertzel Filter Basics

MATLAB

$$\alpha = \frac{2\pi k}{N}$$

$$\beta = \frac{2\pi k(N-1)}{N}$$

$$a = \cos(\beta)$$

$$b = -\sin(\beta)$$

$$c = \sin(\alpha)\sin(\beta) - \cos(\alpha)\cos(\beta)$$

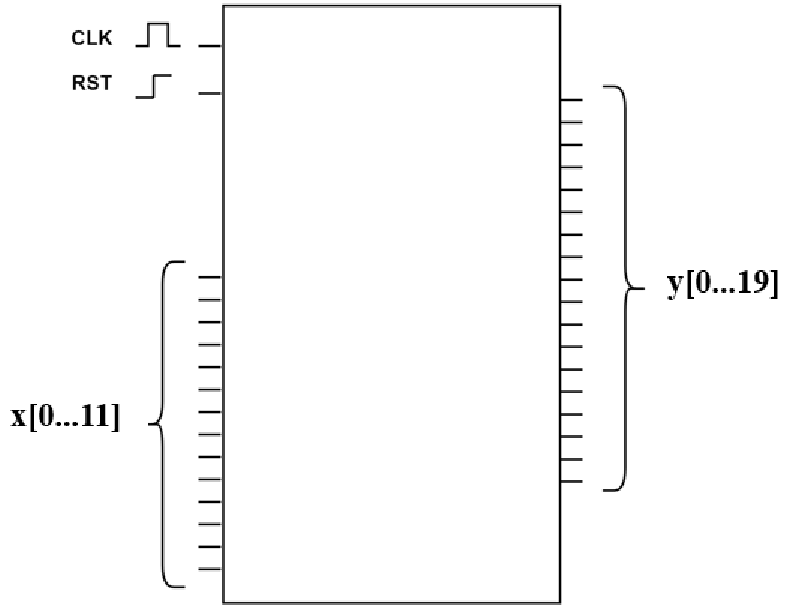
$$d = \sin(2\pi k)$$

$$\omega_0 = \omega_1 = \omega_2 = 0$$

```
k = round(N * f0/fs) ; %frequency index(k=6)
alpha = 2 * pi * k / N;
beta = 2 * pi * k * (N-1)/N;
a = cos(beta)
b = -1 * sin(beta)
c = sin(alpha) * sin(beta) - cos(alpha) * cos(beta)
d = sin(2 * pi * k)
```

VHDL Implementation- Design

```
entity goertzel is
port (
    clk: in std_logic;
    rst: in std_logic;
    x : in unsigned(11 downto 0);
    y : out signed(19 downto 0);
);
end entity;
```



VHDL - Implementation- Design

Architectural Behaviour of Goertzel

```
architecture behav of goertzel is

  -- internal constant signals that are 20 bit signed
  signal a : signed(19 downto 0) := to_signed(973, 20);          -- calculated to be 0.9730
  signal b : signed(19 downto 0) := to_signed(309, 20);          -- calculated to be 0.309017
  signal c : signed(19 downto 0) := to_signed(-1000, 20);        -- calculated to be -1.0
  signal d : signed(19 downto 0) := to_signed(0, 20);            --calculated to be -1.22465e-15
  signal twocosalpha : signed(19 downto 0) := to_signed(1946, 20);--calculated to be 1.946

  -- internal signals for intermediate values that are 20 bit signed
  signal w0 : signed(19 downto 0):= to_signed(0, 20);
  signal w1 : signed(19 downto 0):= to_signed(0, 20);
  signal w2 : signed(19 downto 0):= to_signed(0, 20);
  signal y_real: signed(19 downto 0); -- real component of the output
  signal y_imag: signed(19 downto 0); -- imaginary component of the output

  -- internal signal for counter that is 20 bit signed
  signal counter : signed(19 downto 0):= to_signed(0, 20);
```

Fachhochschule Dortmund
University of Applied Sciences and Arts

Created by Ali

VHDL Implementation- Design

```
begin
    y <= resize(resize((y_real*y_real),y'length) +  resize((y_imag*y_imag),y'length), y'length);

process(clk,rst)
    begin
    if rst'event and rst='1' then --- resetting all signals to 0
        counter <= to_signed(0, counter'length);
        w0 <=      to_signed(0, w0'length);
        w1 <=      to_signed(0, w1'length);
        w2 <=      to_signed(0, w2'length);
        y_real <=  to_signed(0, y_real'length);
        y_imag <=  to_signed(0, y_imag'length);

    elsif clk'event and clk='1' then --on rising clock edge
        if counter < to_signed(134,counter'length) then --if counter is less than 134 then continue with the goertzel algorithm
            w0 <= resize(signed("0000"&x) - to_signed(500,w0'length) +
            resize((twocosalpha*w1)/1000,w0'length)- w2,w0'length);
            counter <= counter + to_signed(1, 20);
        else  -- if counter is greater that 134 then calculate the real and imaginary components of the output and reset the counter.
            y_real <= resize(
            resize(
            resize( a*w1/1000 ,y_real'length) +
            resize( c*w2/1000 ,y_real'length),
            y_real'length) /to_signed(135,y_real'length),
            y_real'length);
            y_imag <= resize(
            resize(
            resize( b*w1/1000 ,y_imag'length) +
            resize( d*w2 ,y_imag'length) ,
            y_imag'length)/to_signed(135,y_imag'length),
            y_imag'length);
            counter <= to_signed(0, 20);
        end if;
    end if;
        w2 <= w1;
        w1 <= w0;
    end process;

end architecture;
```

for $window_{index} = 0$ *to* $window_{size} - 1$ **do**

$$\omega_0 = x_{in} + 2\cos(\alpha)\omega_1 - \omega_2$$

$$\omega_2 = \omega_1$$

$$\omega_1 = \omega_0$$

$$X_k = a\omega_1 + c\omega_2 + j(b\omega_1 + d\omega_2)$$

$$|X_k| = \sqrt{(a\omega_1 + c\omega_2)^2 + (b\omega_1 + d\omega_2)^2}$$

$$\Phi(X_k) = \arctan\left(\frac{b\omega_1 + d\omega_2}{a\omega_1 + c\omega_2}\right)$$

Fachhochschule Dortmund
University of Applied Sciences and Arts

Created by Ali

Verification

Fachhochschule Dortmund
University of Applied Sciences and Arts

MATLAB - Generate 12-bit offset binary data with phase shift

```
sine_freq = [150e3 149e3 151e3 5e3 200e3] ; % Different frequencies for sine waves

phases = [0 30 45 90 120] ; % Different phases in radians
```

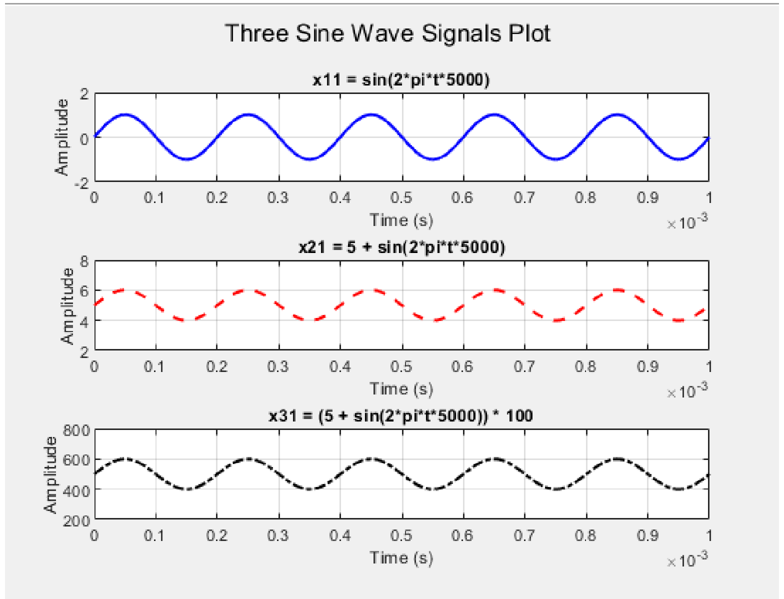
```
for i = 1:length(sine_freq)
    for j = 1:length(phases)
    data(:,1) = round((sin(2*pi*sine_freq(i) *t + phases(j)* pi/180) + 5) *100);
    filename = 'sine_'+string(sine_freq(i)) + 'KHz_' +string(phases(j)) + 'deg.txt';
    dlmwrite(filename, data(:,1), ' ');
    end
end
```

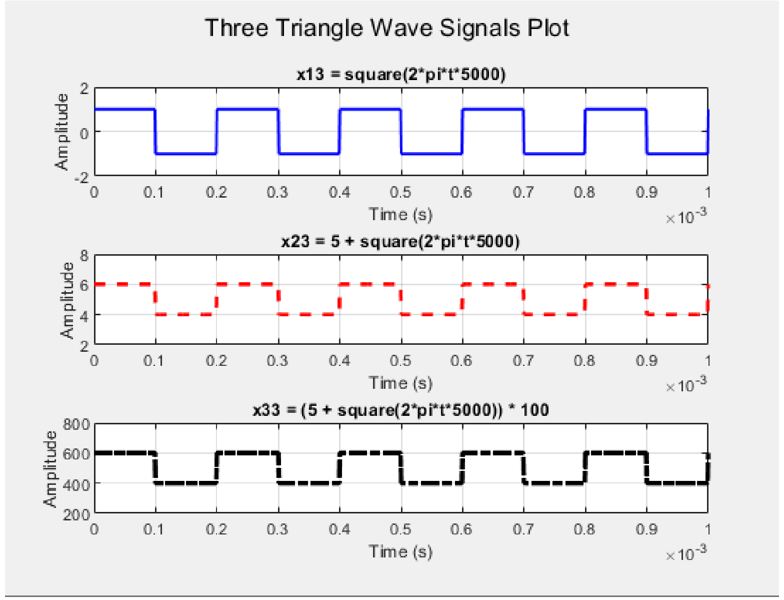
Fachhochschule Dortmund

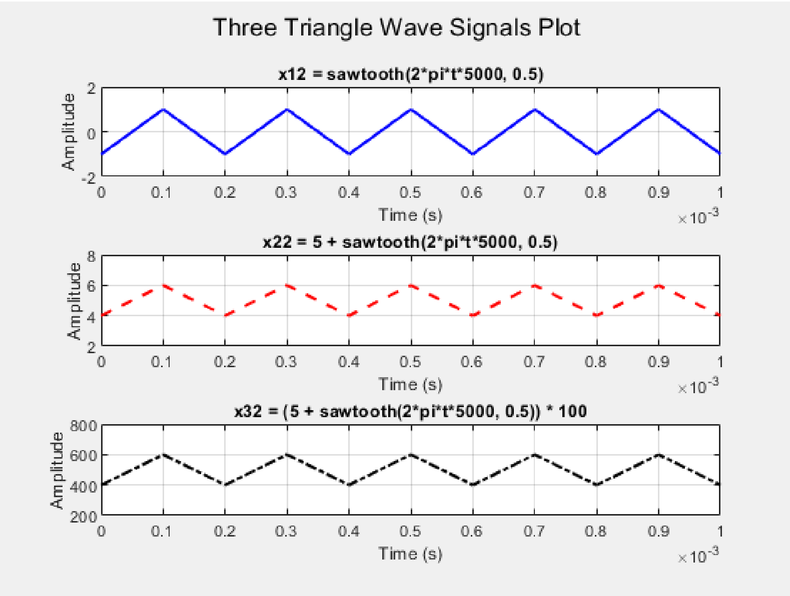
University of Applied Sciences and Arts

Created by Ghazaleh & Roghieh

Scaled and Shifted Input Signal







Fachhochschule Dortmund
University of Applied Sciences and Arts

Created by Ghazaleh & Roghieh

VHDL TEST BENCH

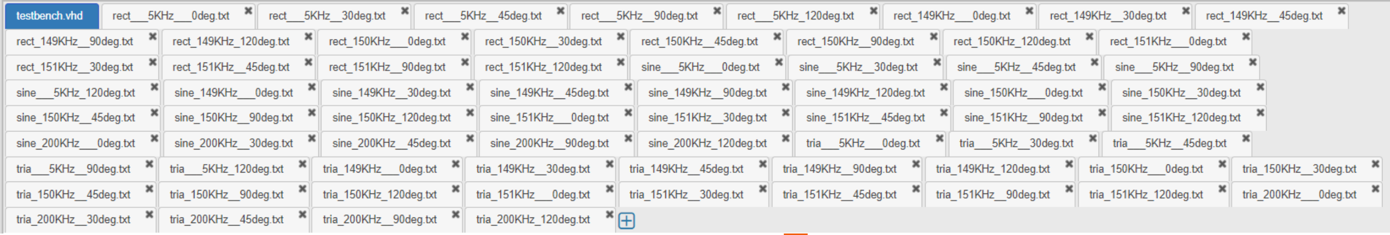
```vhdl
-- Signal declarations
signal   rst      :  std_logic := '0';
signal   clk      :  std_logic := '0';
signal   x        :  unsigned(11 downto 0);
signal   y        :  signed(19 downto 0);
signal   fileName : string(1 to 22) := to_string("sine_149KHz_120deg.txt"); --- sample f

begin -- architecture

  -- Instantiate the DUT (Device Under Test)
  dut: goertzel port map (
      rst => rst,
      clk => clk,
      x   => x,
      y   => y
      );
  process
      file testFile : text;
      variable lineStr : line;
      variable readVal : integer;
      variable count   : natural := 0;
  begin
      file_open(testFile, fileName, read_mode);
      while not endfile(testFile) loop
          readline(testFile, lineStr);
          read(lineStr, readVal);
          x <= unsigned(to_unsigned(readVal, x'length));
          clk <= '0';
          wait for 1 ns;
          clk <= '1';
          wait for 1 ns;
          count := count + 1;
          if count = 135 then
            report  fileName&" -> power "& to_string(to_integer(y)) &"-> Amplitude: " & to_string(sqrt(real(to_integer(y))),"%f") & ".";
            exit;
          end if;

       end loop;

      file_close(testFile);
      wait;
  end process;
```



Test Cases are:

- Sine Waves with 150 kHz, 149 kHz, 151 kHz, 5 kHz, 200 kHz
- Square Waves with 150 kHz, 16 kHz, 10 kHz, 200 kHz
- Triangle wave with 150 kHz, 149 kHz, 151 kHz, 5 kHz, 200 kHz
- For all of the above alter phase angles: 0°, 30°, 45°, 90°, 120°

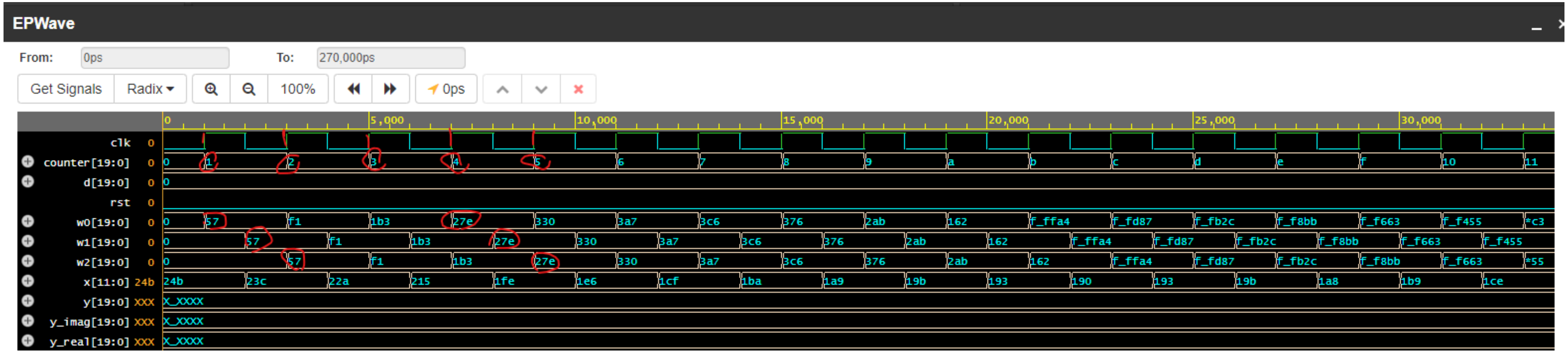
Fachhochschule Dortmund

University of Applied Sciences and Arts

Created by Ali

EPWave Simulation

```
x <= unsigned(to_unsigned(readVal, x'length));
clk <= '0';
wait for 1 ns;
clk <= '1';
wait for 1 ns;
```

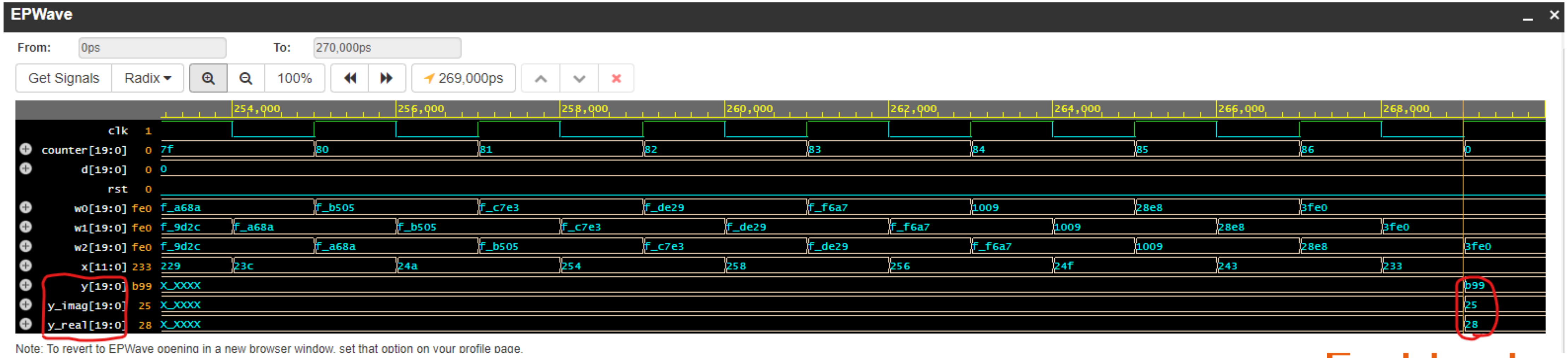


Fachhochschule
Dortmund
University of Applied Sciences and Arts

Created by Ghazaleh & Roghieh

EPWave Simulation

```vhdl
x <= unsigned(to_unsigned(readVal, x'length));
clk <= '0';
wait for 1 ns;
clk <= '1';
wait for 1 ns;
```



Fachhochschule Dortmund
University of Applied Sciences and Arts

Created by Ghazaleh & Roghieh

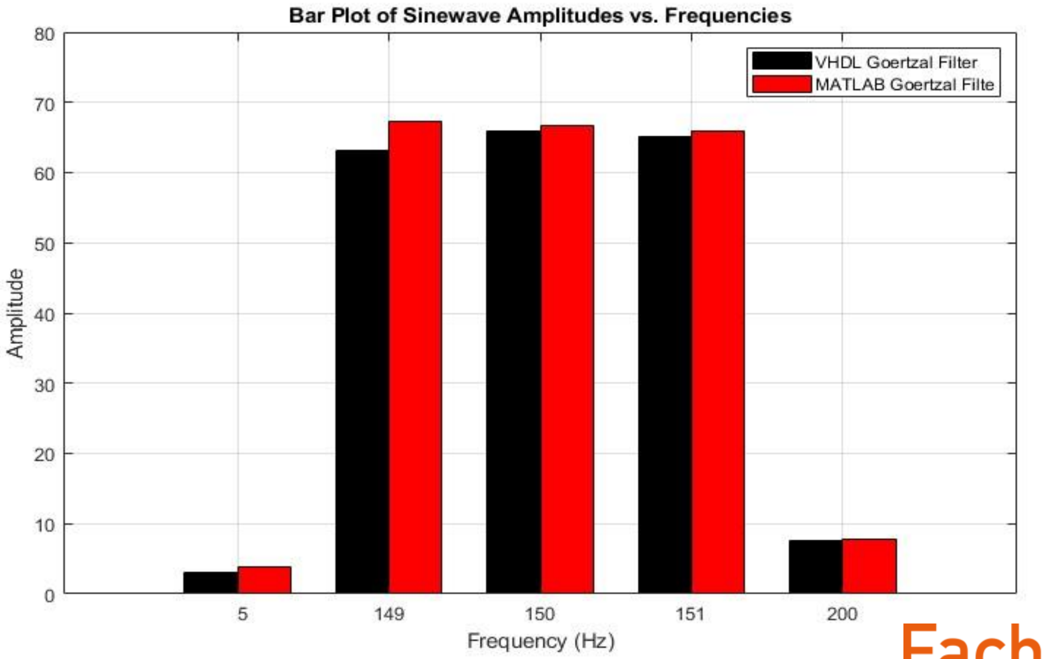
Result Comparison

150 KHZ SIGNAL DETECTION OF SINEWAVE TESTCASES

```
% Parameters
N = length(data); % Number of samples
sample_rate = 4e6; % Sample rate in Hz (4 MHz)
frequency_target = 151e3; % Target frequency in Hz
scale_factor = 100; % Scaling factor for conversion
% Compute the frequency bin index for the target frequency
k_target = round(frequency_target * N / sample_rate)+1;

% Apply Goertzel filter
goertzel_result = goertzel(data, k_target);

magnitude = abs(goertzel_result);
fprintf('Goertzel magnitude: %.3f\n', magnitude );
```

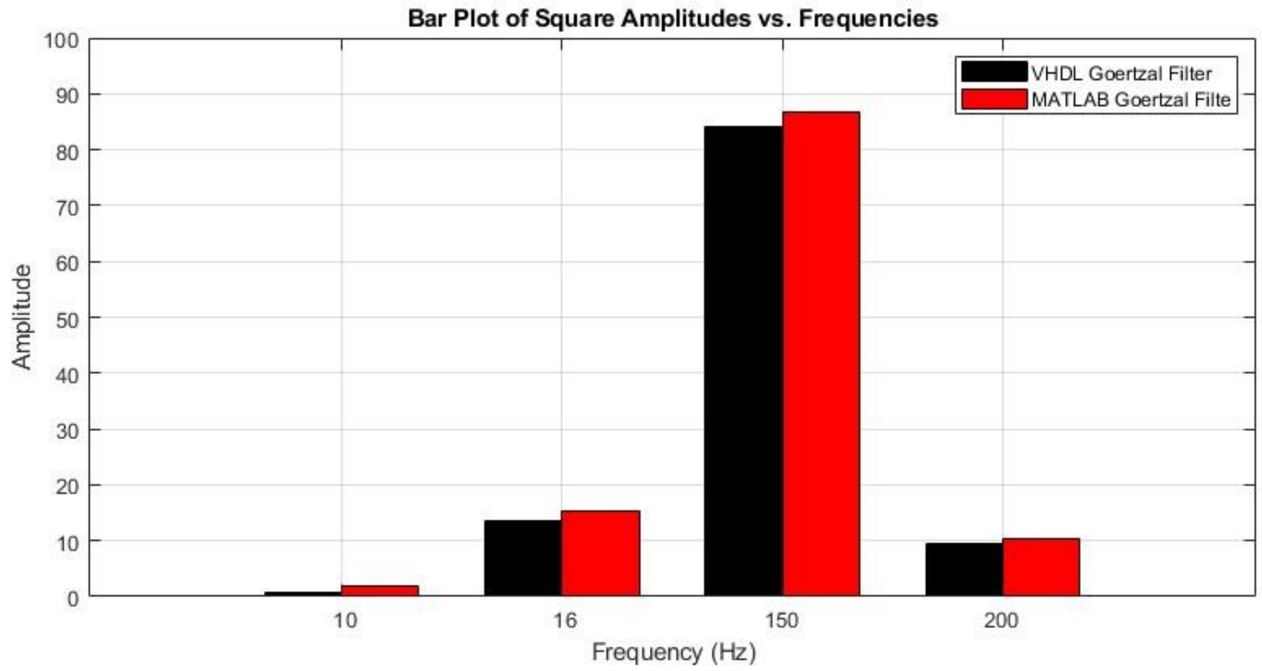


Fachhochschule Dortmund
University of Applied Sciences and Arts

Created by Ghazaleh & Roghieh

Result Comparison

150 KHZ SIGNAL DETECTION OF SQUAREWAVE TESTCASES



Fachhochschule Dortmund
University of Applied Sciences and Arts

Created by Ghazaleh & Roghieh

Conclusion

- The Goertzel Algorithm is a Digital Signal processing technique used for efficient and fast calculation of individual DFT coefficients.

- It provides a computationally efficient alternative to the Fast Fourier Transform (FFT) algorithm when the calculation of a specific frequency component is required rather than the entire spectrum. It is widely used in Dual-tone multi frequency(DTFM) application.

References

- Petr Sysel and Pavel Rajmic. **“Goertzel algorithm generalized to non-integer multiples of fundamental frequency.”** EURASIP Journal on Advances in Signal Processing 2012.
- L. P. Ferreyro, M. García Redondo, M. R. Hampel, A. Almela, A. Fuster, J. Salum, J. M. Geria, J. Bonaparte, J. Bonilla-Neira, N. Müller, N. Karcher, O. Sander, M. Platino, M. Weber, A. Etchegoyen. **“An Implementation of a Channelizer based on a Goertzel Filter Bank for the Read-Out of Cryogenic Sensors.”** Journal of Instrumentation.