```
function [quantized_signal] = quantizer(sampled_signal,varargin)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% (Uniform mid-rise quantizer)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Sample of input for quantizer funtion:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% ts = 0.1;
%%% nLevels = 5;
%%% mp = 5;
%%% m_law=2;
%%% [binary_signal,level_signal,quantized_signal] = quantizer(sampled_sig,'NLevels',
nLevels, 'SigMax', mp, 'QuantizerType', 0,'MeuValue',m_law);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Input Oarsing Handeling
quantizationType = 1;
mp = max(sampled_signal);
nLevels = 4;
meu = 1;

p = inputParser();
addOptional(p, 'QuantizerType', quantizationType, @isnumeric);
addOptional(p, 'NLevels', nLevels, @isnumeric);
addOptional(p, 'MeuValue', meu, @isnumeric);
addOptional(p, 'SigMax', mp, @isnumeric);
parse(p, varargin{:});

nLevels = p.Results.NLevels;
mp = p.Results.SigMax;


if (2^(ceil(log2(nLevels))) > nLevels)
    disp('Number of Levels must be multiple of 2');
    nLevels = 2^(ceil(log2(nLevels)));
    fprintf('A %d number of levels was chosen instead \n',nLevels);

end

%% Uniform mid-rise quantizer

quantized_signal = zeros(size(sampled_signal));
level_signal= zeros(size(sampled_signal));
detla = 2*mp/(nLevels-1);

for n =1:length(sampled_signal)
    current_level = -mp;
    level_number = 0;
    for k= 1:nLevels

        if((sampled_signal(n) <= current_level && sampled_signal(n) >= current_level -
detla/2) || (sampled_signal(n) >= current_level && sampled_signal(n) <= current_level
+ detla/2))
            quantized_signal(n) = current_level;
            level_signal(n) = level_number;
            break;
        end

        level_number = level_number + 1;
        current_level = current_level + detla;

    end
end
end
```