FFT - Fast Fourier Transform Iterative Implementation

Fast Re-ordering of Samples

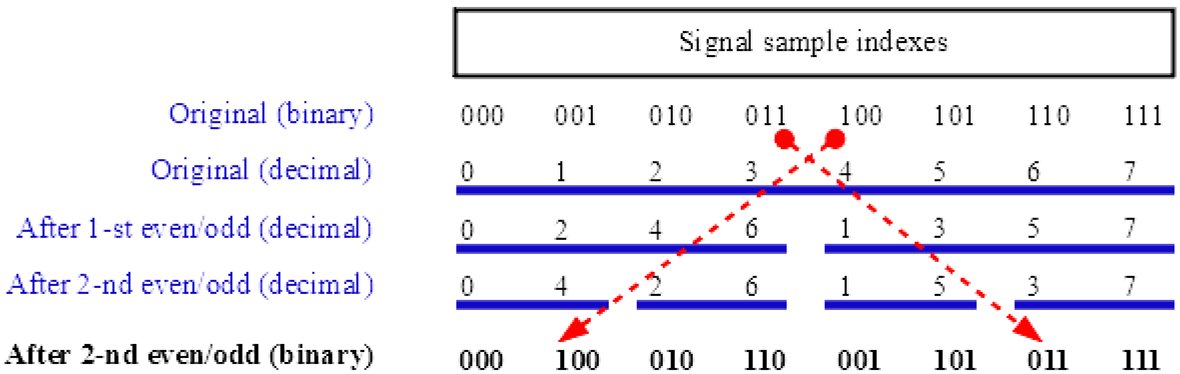
In the recursive FFT implemtation signal samples are even/odd partitioned many times. In contrary, in an iterative FFT algorithm implementation each signal sample comes directly from its input position to the final position which is the same as after multi-level sample partitions:

the sample index is written as an unsigned integer number in binary system and **position of all bits is left/right reversed** (now they are written from LSB to MSB).

See figure below descring this procedure for $N = 8$. For example (**old** sample number → **new** sample number):

3 = 011b → 110b = 6

4 = 100b → 001b = 1.



2-Point DFTs

After samples re-ordering, $N/2$ 2-point DFTs are performed upon pairs of samples, in the figure above upon the following pairs:

$$[x(0), x(4)], \quad [x(2), x(6)], \quad [x(1), x(5)], \quad [x(3), x(7)].$$

The 2-sample DFT is defined as ($y(n)$ - input, $Y(k)$ - output):

$$\begin{bmatrix} Y(0) \\ Y(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & e^{-j\pi} \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \end{bmatrix}, \qquad \begin{cases} Y(0) = y(0) + y(1) \\ Y(1) = y(0) - y(1) \end{cases} .$$

and it consists of only one addition and one subtraction.

Iterative DFT spectra combining

Then 2-point spectra are combined into 4-point ones, 4-point into 8-point, and so on, according to the general spectrum accumulation equation:

$$X^{(K)}(k) = \left[X_e^{(K/2)}(k_1),\ X_e^{(K/2)}(k_1)\right] + e^{-j\frac{2\pi k}{K}} \cdot \left[X_o^{(K/2)}(k_1)\ \ X_o^{(K/2)}(k_1)\right],$$

where:

- $K = 4, 8, 16, ..., N$ - length of the one combined K-samples long spectrum $X^{(K)}(k)$,
- $k = 0, 1, ..., K-1$ - indexes of the combined spectrum,
- $k_1 = 0, 1, ..., K/2-1$ - indexes of even/odd $K/2$-samples long spectra $X_{e/o}^{(K/2)}(k_1)$ being combined, two times shorter than $X^{(K)}(k)$.

Testing

Analyze the iterative Radix-2 DIT FFT function presented below. Run the program. Change signal length and values of its samples.

```
clear all;                       % clear all variables
N = 8; x = rand(1,N);            % analyzed signal, N=2^p
Xm = fft(x);                     % its DFT (FFT) computed by Matlab
X = myIterFFT(x);                % calling recursive function
error = max( abs( X - Xm ) ),    % error
```

```
error = 2.7336e-16
```

```
function y = myIterFFT(x)
% My non-recursive radix-2 FFT function
N = length(x);        % number of signal samples
Nbits = log2(N);      % number of bits for sample indexes, e.g. for N=8, Nbit=3

% Samples reordering (in bitreverse fashion)
% x = 0:N-1;            % only for test
  n = 0:N-1;            % indexes of ALL samples
  m = dec2bin(n);       % bits of these indexes
  m = m(:,Nbits:-1:1);  % reverse of these bits
  m = bin2dec(m);       % new indexes of ALL samples
  y(m+1) = x(n+1);      % data reordering
% y, pause              % checking re-ordering result

% ALL 2-point DFTs upon the neighboring pairs of partitioned (sorted) data
  y = [ 1 1; 1 -1] * [ y(1:2:N); ...
                       y(2:2:N) ]; y = y(:)';    % 2-point DFT spectra

% N-point DFT spectrum reconstruction
% DFT spectra: 2-point --> 4-point --> 8-point --> 16-point ...
  Nx = N;                   % number of samples (changing variable length)
  Nlevels = Nbits;          % number of levels of computations equal to log2(Nx)
```

```
  N = 2;                    % initial DFT length after 2-point DFTs
  for lev = 2 : Nlevels                % next LEVEL
      N = 2*N;                         % new DFT spectrum length after combining
      Nblocks = Nx/N;                  % number of new DFT spectra after combining
      W = exp( -j*2*pi/N*(0:N-1) );    % correction term on this level
      for k = 1 : Nblocks                                  % next BLOCK of 2 even/odd DFTs
          y1 = y( 1     + (k-1)*N : N/2 + (k-1)*N );       % y1 even (LOWER) spectrum
          y2 = y( N/2+1 + (k-1)*N : N   + (k-1)*N );       % y2 odd  (UPPER) spectrum
          y(1 + (k-1)*N : N + (k-1)*N ) = [ y1 y1 ] + W .* [ y2 y2 ];  % combining y1,y1 & y2,y
      end
  end
end
```