Szybkie przekształcenie Fouriera

ang. *Fast Fourier Transform*

FFT

FFT jest szybką implementacją DFT – dyskretnej transformacji Fouriera .

FFT wykorzystuje się w analizie widmowej sygnałów/systemów o skończonej liczbie próbek i/lub do implementacji – za pomocą splotu kołowego – systemów o skończonej odpowiedzi impulsowej (ang. *finite impulse response* – FIR), ale **nie** do implementacji systemów o nieskończonej odpowiedzi impulsowej (ang. *infinite impulse response* – IIR).

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{\,nk},\ k=0,1,\cdots,N-1 \quad \text{DFT wzór analizy (*)}$$

$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X[k]W_N^{\,-kn},\ n=0,1,\cdots,N-1 \quad \text{IDFT wzór syntezy}$$

Z równania (*) wynika, że bezpośrednie obliczenie z niego N próbek DFT wymaga $N(N-1)$ mnożeń i podobnej liczby dodawań.

$$[x_N]\stackrel{\Delta}{=}\begin{bmatrix}x[0]\\x[1]\\\vdots\\x[N-1]\end{bmatrix} \qquad [X_N]\stackrel{\Delta}{=}\begin{bmatrix}X[0]\\X[1]\\\vdots\\X[N-1]\end{bmatrix}$$

$$[X_N]=[W_N][x_N] \qquad W_N\stackrel{\Delta}{=}e^{-j\frac{2\pi}{N}}$$

$$[x_N]=[W_N]^{-1}[X_N]=\frac{1}{N}[W_N^*][X_N]$$

$$[W_N]=\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1}\\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)}\end{bmatrix}=[W_N]^T$$

$[\ ]^T$ transpozycja macierzy

Metody FFT wykorzystują właściwości okresowości i symetrii podstawowej funkcji Fouriera – czynnika obrotu $W_N \triangleq e^{-j\frac{2\pi}{N}}$ (ang. *twiddle factor*), by uniknąć powtarzania niektórych obliczeń. W szczególności, metody FFT wykorzystują:

1. Okresowy charakter $W_N^{nk} = W_N^{(n+N)k} = W_N^{n(k+N)}$.
2. Symetrię zespoloną sprzężoną $W_N^{-nk} = \left(W_N^{nk}\right)^* = W_N^{(N-n)k}$.

Algorytmy wykorzystujące rozłożenie ciągu $x[n]$ na coraz krótsze podciągi są nazywane algorytmami z podziałem czasowym (ang. *decimation-in-time FFT*), a algorytmy wykorzystujące rozłożenie ciągu $X[k]$ na coraz krótsze podciągi są nazywane algorytmami z podziałem częstotliwościowym (ang. *decimation-in-frequency FFT*). Są one głównie przeznaczone do realizacji, gdy N jest potęgą liczby 2, czyli $N = 2^r$ (ang. *radix-2 FFT*).

Algorytm z podziałem czasowym

Ciąg N próbek $\{x[n], \quad n = 0,1,\cdots,N-1\}$

dekomponujemy na dwa krótsze podciągi: o numerach parzystych

$$x_1[n] = x[2n], \quad n = 0,1,\cdots,N/2-1$$

i o numerach nieparzystych

$$x_2[n] = x[2n+1], \quad n = 0,1,\cdots,N/2-1$$

DFT można rozłożyć na dwie DFT o długości $N/2$ każda. Tzn.

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} = \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{(2n+1)k}$$

Ponieważ $W_N^{2nk} = e^{-j\frac{2\pi}{N}2nk} = e^{-j\frac{2\pi}{N/2}nk} = W_{N/2}^{nk}$, to poprzednie równanie można zapisać następująco

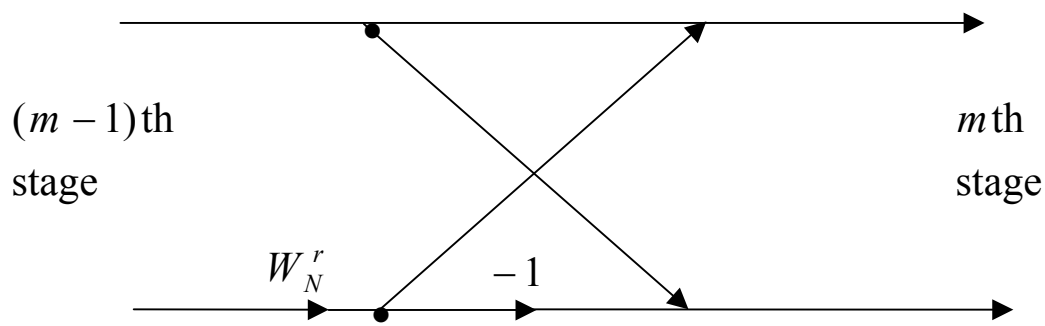
$$X[k] = \sum_{n=0}^{N/2-1} x_1[n] W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x_2[n] W_{N/2}^{nk} \qquad (**)$$

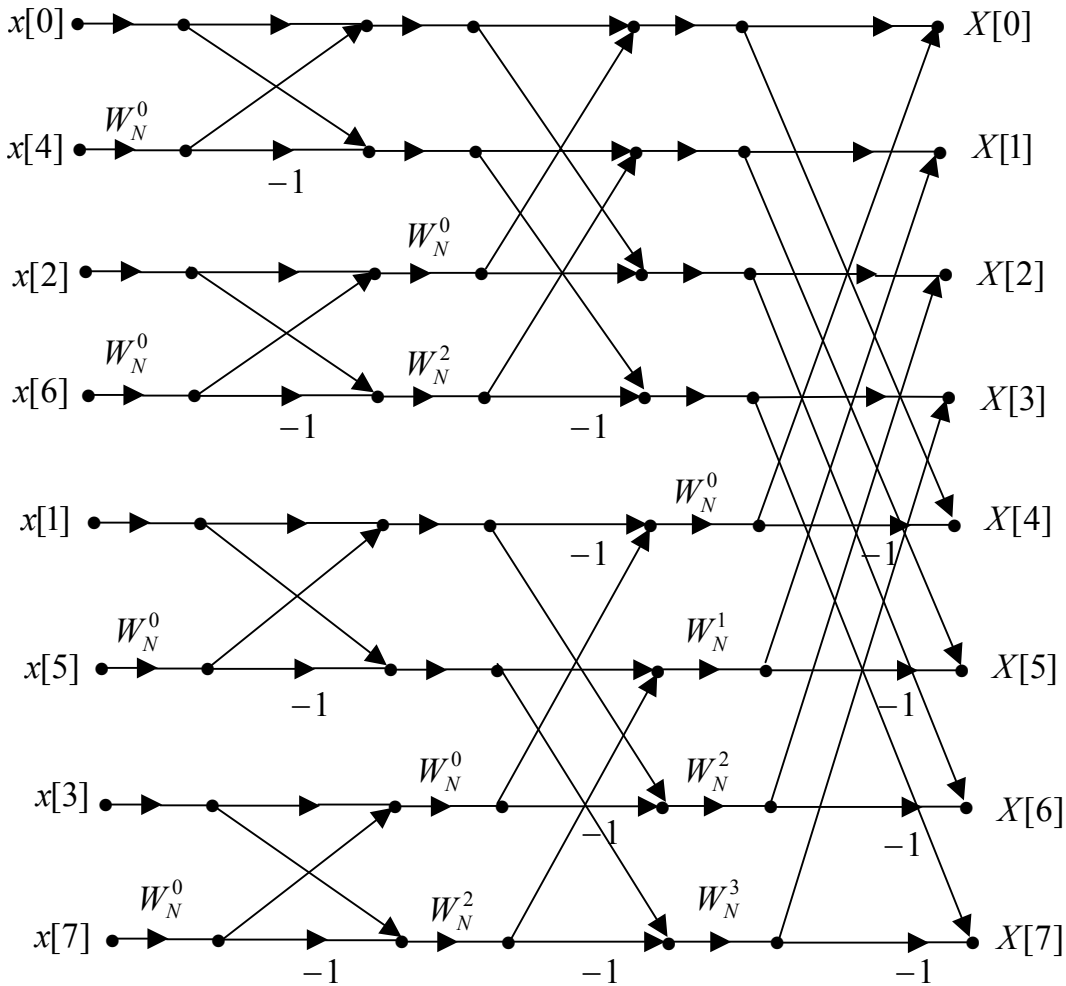
W (**) każdy składnik sumy redukuje się do $N/2$ punktowej DFT. Ponadto, na podstawie warunków symetrii i okresowości, równanie (**) można zapisać jako

$$X[k] = X_1[k] + W_N^k X_2[k], \quad k = 0,1,\cdots,N-1$$

$$= \begin{cases} X_1[k] + W_N^k X_2[k], & k = 0,1,\cdots,N/2-1 \\ X_1[k] - W_N^k X_2[k], & k = N/2,\cdots,N-1 \end{cases}$$

gdzie $X_1[k] = \mathrm{DFT}\{x_1[n]\}$ i $X_2[k] = \mathrm{DFT}\{x_2[n]\}$ wykorzystując $N/2$-punktową DFT. To równanie reprezentuje obliczenia motylkowe (ang. *butterfly computation*). Ten proces powtarza się, aż skończymy na zbiorze 2-punktowych DFT, ponieważ N jest potęgą liczby 2.





Graf przepływowy algorytmu 8-punktowej DFT z podziałem czasowym (Cooley-Tukey).

Złożoność numeryczna algorytmu 2-punktowej FFT z $N = 2^r$ gdzie r jest liczbą całkowitą

Założenie: ciąg $x[n]$ ma próbki o wartościach rzeczywistych. Przetwarzanie blokowe: blok N próbek $x[n]$ jest transformowany w blok N próbek $X[k]$.

DFT – $N(N-1)$ zespolonych mnożeń i $N(N-1)$ zespolonych dodawań,

FFT – $(N/2)\log_2(N/2)$ zespolonych mnożeń i $N\log_2 N$ zespolonych dodawań.

Dla $N = 256$ wyniki dla bezpośredniej **DFT:**

256X255= 65 280 zespolonych mnożeń,

256X255= 65 280 zespolonych dodawań.

Za pomocą **FFT** złożoność numeryczna jest znacznie zredukowana do:

128X $\log_2$ 128= 896 zespolonych mnożeń,

256X $\log_2$ 256= 2 048 zespolonych dodawań.

Odwracanie kolejności bitów (ang. *bit reversal*)

FFT wymaga pogrupowania razem próbek o indeksach parzystych i o indeksach nieparzystych.

Indeks czasowy próbek sygnału wejściowego w formacie binarnym ma odwróconą kolejność bitów względem porządku naturalnego. Przykładowo, próbka x[100] z binarnym zapisem indeksu, czyli x[4], jest przechowywana na pozycji x[001], tj. x[1], i x[110], tj. x[6], jest przechowywana na pozycji x[011], tj. x[3], itd.

Nowoczesne procesory DSP takie, jak TMS320C55x mają sprzętową realizację odwracania kolejności bitów. Dzięki temu można przechowywać w pamięci procesora blok próbek sygnału wejściowych z odwróconą kolejnością bitów zapewnianą hardware'owo.

Odwrotna FFT – IFFT (ang. *Inverse FFT*)

IFFT otrzymuje się przez zastąpienie zmiennej W (czynnik obrotu) przez jej odwrotność i skalowanie przez czynnik N jak pokazano na slajdzie No 2.

Obliczenia z podstawianiem (ang. *in-place computations*)

FFT wymaga zapamiętywania jedynie N wartości zespolonych.

Algorytmy FFT wykonujące obliczenia z podstawianiem wykorzystują te same lokalizacje, w pamięci bloku próbek wejściowych, wszystkich wyników obliczeń pośrednich i bloku próbek wyjściowych.

Implementacje w środowisku MATLAB

MATLAB zawiera funkcję

Y=fft(x);

do obliczania DFT of ciągu czasowego zapisanego w wektorze x. Jeżeli x jest macierzą, to y jest transformatą DFT każdej kolumny tej macierzy. Jeżeli długość x jest potęgą liczby 2, to funkcja fft wykorzystuje bardzo szybki algorytm radix-2 FFT. W przeciwnym przypadku stosowany jest wolniejszy algorytm mieszany (ang. *mixed-radix* FFT).

Alternatywnym sposobem wykorzystania funkcji fft jest

Y=fft(x,N);

który wykorzystuje N – punktową FFT pierwszych N próbek wektora x.

Algorytm z podziałem częstotliwościowym

Algorytm FFT z podziałem częstotliwościowym jest bardzo podobny do algorytmu z podziałem czasowym. W reprezentacji motylkowej właściwości symetrii są odwrócone względem algorytmu z podziałem czasowym. Odwrócona kolejność bitów występuje na wyjściu zamiast na wejściu. Próbki wyjściowe są uporządkowane wg odwróconej kolejności bitów indeksu próbki w zapisie binarnym. Złożoność numeryczna algorytmu FFT z podziałem częstotliwościowym jest taka sama, jak odpowiadającego mu algorytmu FFT z podziałem czasowym.

Zobacz przykładowo:

A.V. Oppenheim, R.W. Schafer with J.Buck: Discrete-Time Signal Processing, Prentice Hall, 1999, Chapter 8 (dostępna w bibliotece WETI).

Splot liniowy ciągu o skończonej długości z ciągiem bardzo długim realizowany za pomocą splotu kołowego – DFT/FFT – **metoda sumowania z nakładaniem** (ang. overlap-add)

Rozparzmy splot sygnału $x[n]$ z ciągiem $h[n]$ o długości M. Rozłóżmy ciąg $x[n]$ o długości nieskończonej na segmenty zawierające tylko L wyrazów. Niech k-ty segment będzie oznaczony symbolem $x_k[n]$

$$x_k[n] = \begin{cases} x[n], & kL \le n \le (k+1)L-1 \\ 0, & \text{w przeciwnym przypadku} \end{cases}$$

Zatem ciąg $x[n]$ jest równy sumie ciągów $x_k[n]$, czyli

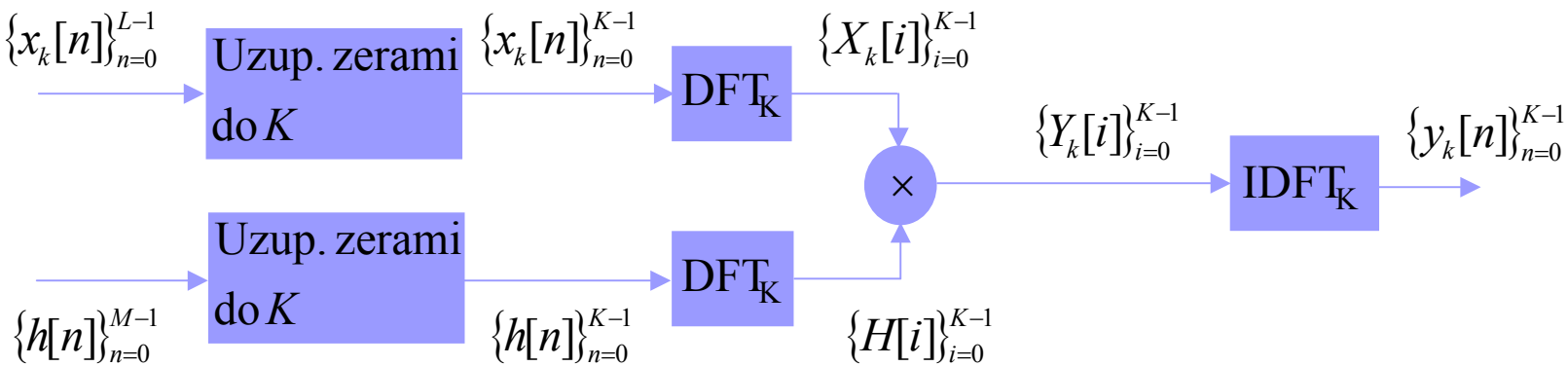
$$x[n] = \sum_{k=0}^{\infty} x_k[n]$$

A splot liniowy $y[n]$ ciągu $x[n]$ z ciągiem odpowiedzi impulsowej $h[n]$ jest równy sumie splotów ciągów $x_k[n]$ z odpowiedzią impulsową $h[n]$

$$y[n] = x[n] * h[n] = \sum_{k=0}^{\infty} x_k[n] * h[n] = \sum_{k=0}^{\infty} y_k[n-kL]$$

Ponieważ ciąg $x_k[n]$ ma tylko L wyrazów, a $h[n]$ ma długość M, to każdy z członów $y_k[n] = x_k[n] * h[n]$ sumy ma długość L+M-1. Z tego względu splot liniowy $x_k[n] * h[n]$ należy wyznaczać na podstawie (L+M-1)-punktowych DFT.

Przy sumowaniu wystąpi zjawisko zachodzenia na siebie $M-1$ różnych od zera wyrazów segmentów odfiltrowanych. Sygnał wyjściowy można utworzyć sumując odfiltrowane segmenty. Ten sposób tworzenia odfiltrowanego sygnału wyjściowego jest nazywany metodą sumowania z nakładaniem. Alternatywny sposób (patrz literatura na slajdzie No.10) nazywany jest metodą sumowania bez nakładania (ang. *overlap-save*).

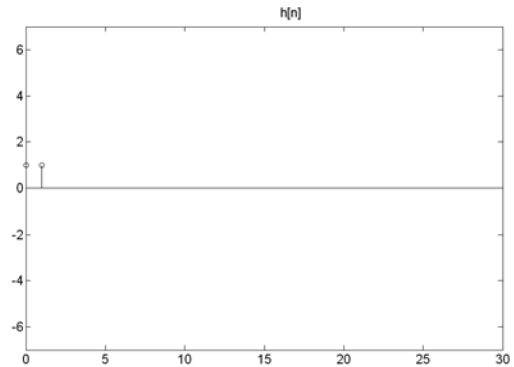


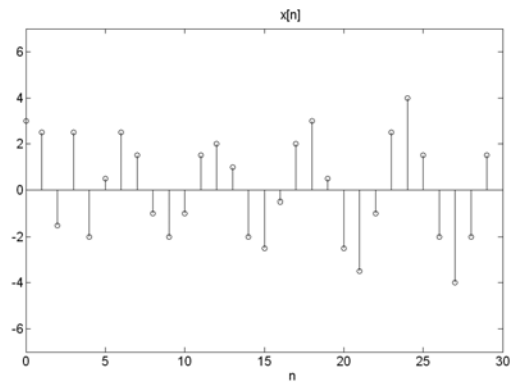
Schemat przetwarzania k -tego segmentu, $K \geq L+M-1$.

Przykład. Ciągi splatane

x=[3 2.5 -1.5 2.5 -2 0.5 2.5 1.5 -1 -2 -1 1.5 2 1 -2 -2.5 -0.5 2 3 0.5 -2.5 -3.5 -1 2.5 4 1.5 -2 -4 -2 1.5];

h=[1 1];

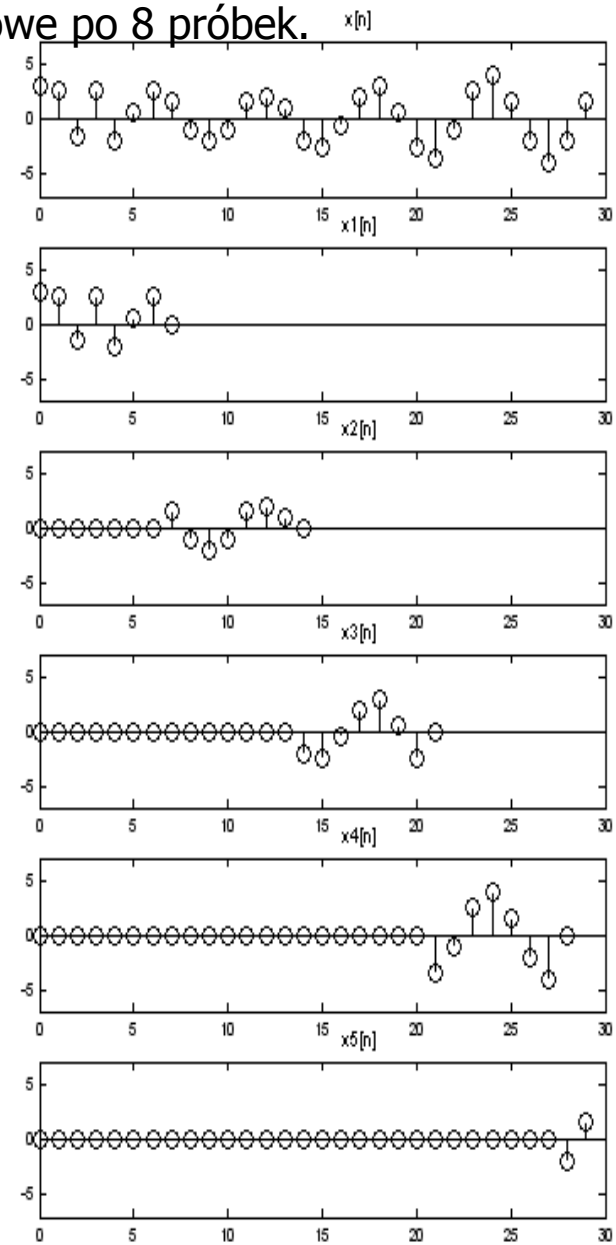


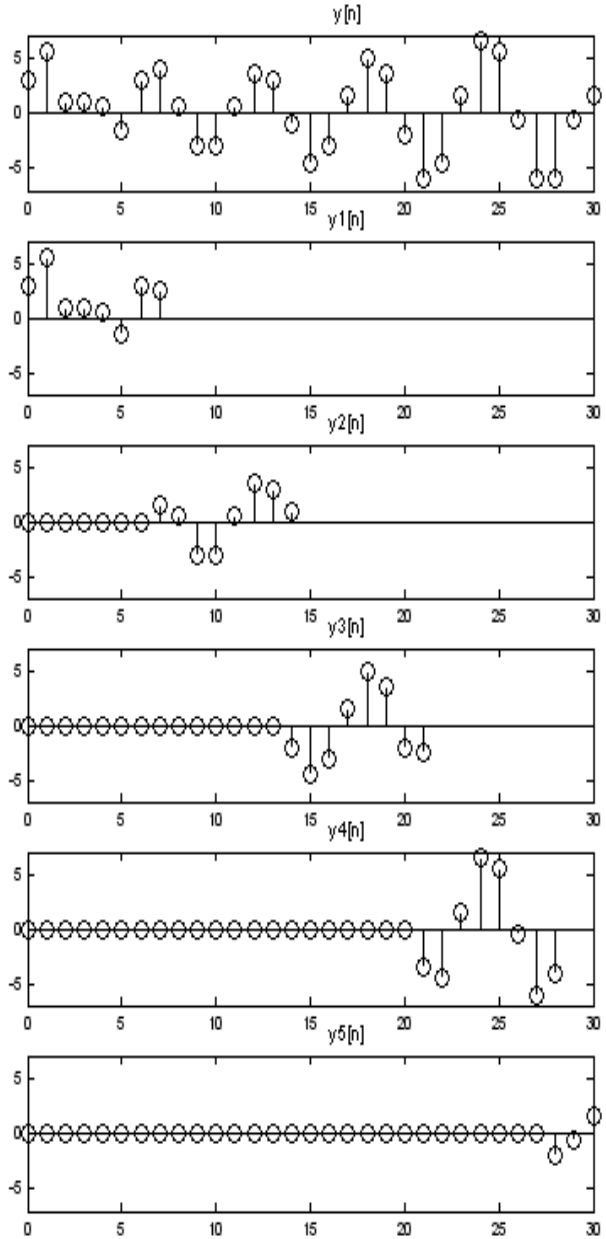


Wynik końcowy splotu

y=[3.0 5.5 1.0 1.0 0.5 -1.5 3.0 4.0 0.5 -3.0 -3.0 0.5 3.5 3.0 -1.0 -4.5 -3.0 1.5 5.0 3.5 -2.0 -6.0 -4.5 1.5 6.5 5.5 -0.5 -6.0 -6.0 -0.5 1.5]

Bloki wejściowe liczą po 7 próbek sygnału wejściowego plus dopisana próbka 0, a bloki wyjściowe po 8 próbek.





$$x[n] = x_1[n] + x_2[n] + x_3[n] + x_4[n] + x_5[n]$$

$$y[n] = y_1[n] + y_2[n] + y_3[n] + y_4[n] + y_5[n]$$

Dodawanie z nakładaniem