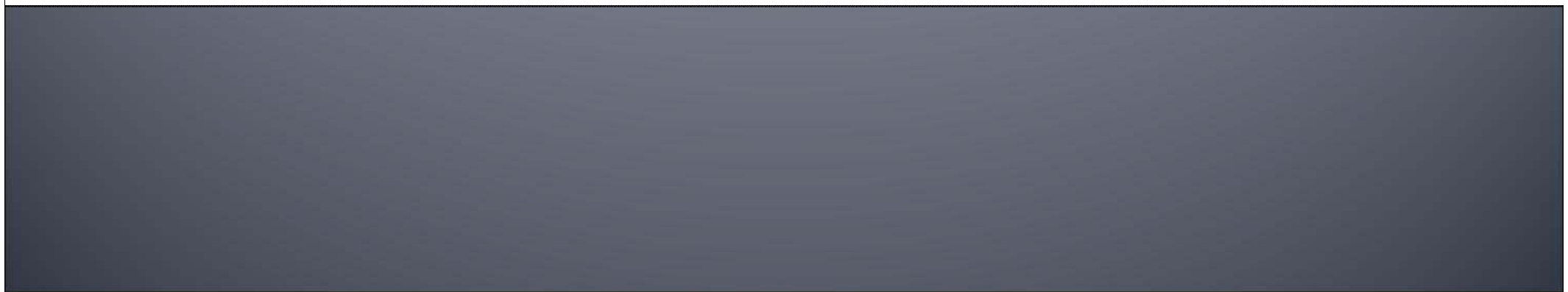
Zastosowania Procesorów Sygnałowych

dr inż. Grzegorz Szwoch
greg@multimed.org
p. 732 - Katedra Systemów Multimedialnych

Przekształcenie Fouriera i splot



Wstęp

Na tym wykładzie:

- przekształcenie Fouriera na DSP
- okienkowanie sygnału
- splot – zwykły i szybki
- filtracja blokowa FIR
- splot ciągłego sygnału

FFT - wprowadzenie

- Transformacja Fouriera: przekształcenie sygnału z dziedziny czasu do dziedziny częstotliwości.
- Transformata Fouriera: wynik transformacji.
- Umożliwia analizę i przetwarzanie sygnału w dziedzinie częstotliwości.
- FFT: szybka TF, algorytm zoptymalizowany obliczeniowo, bardzo dobry dla DSP.
- Procesory sygnałowe mają zwykle sprzętową implementację obliczania FFT.

FFT - podstawy

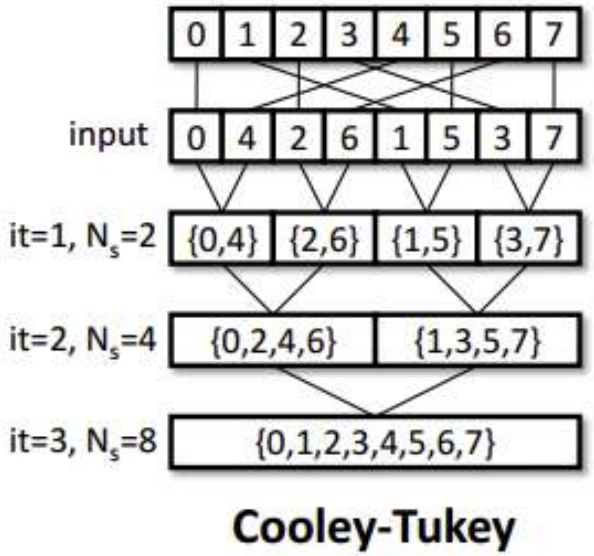
- FFT – *Fast Fourier Transform*
- Dyskretna TF oblicza transformatę ze wzoru – jest wolna.
- FFT iteracyjnie dzieli sygnał „na pół”, aż do otrzymania par, liczy 2-punktowe FFT, z ich wyników ponownie liczy 2-punktowe FFT, aż do uzyskania końcowego wyniku.
- Aby to było możliwe, liczba próbek (rozmiar transformaty) musi być potęgą dwójki: 2^N

FFT – uzupełnianie zerami

- Co zrobić jeżeli nie mamy 2^N próbek?
- Można uzupełnić blok próbek zerami, do najbliższej potęgi dwójki – *zero padding*.
- To nie znaczy że dostaniemy większą dokładność transformaty. Dodatkowe wartości są interpolowane w dziedzinie częstotliwości.
- Wartości FFT po uzupełnieniu zerami będą różne od uzyskanych z DFT bez uzupełnienia. Są to „próbki” widma dla innych punktów na skali częstotliwości.

FFT - dekompozycja

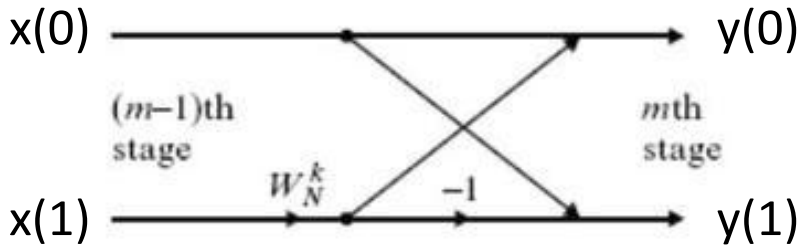
- Rozpatrzmy FFT o rozmiarze 8 (3 etapy).
- Dekompozycja sygnału na dwie sekwencje, według zasady:
  - parzyste indeksy,
  - nieparzyste indeksy



Cooley-Tukey

Dwupunktowa FFT

- FFT o rozmiarze 2 oblicza się następująco:



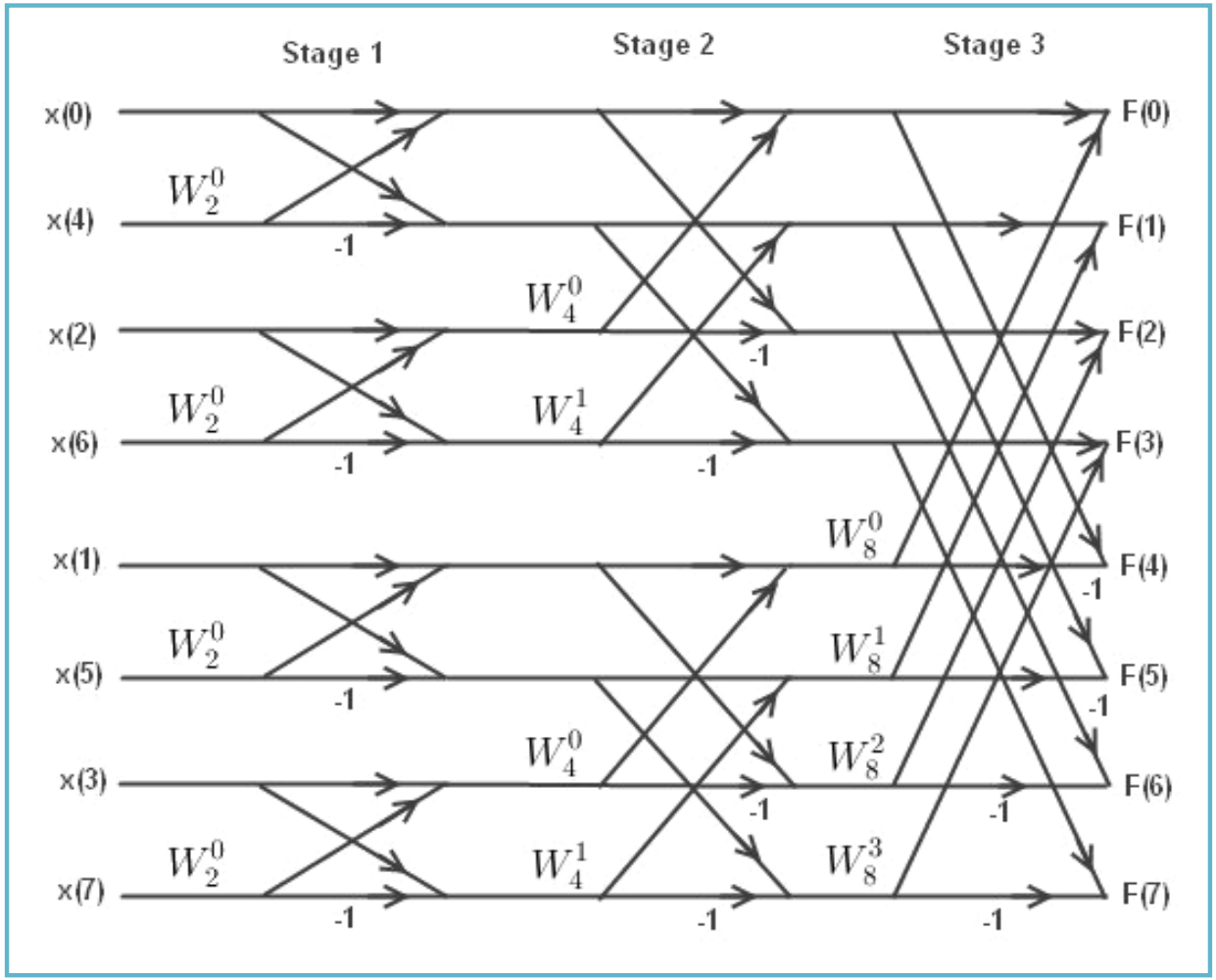
$$y(0) = x(0) + x(1) \cdot W_n^k$$

$$y(1) = x(0) - x(1) \cdot W_n^k$$

- Taką strukturę nazywa się *motylkiem*.
- Tylko jedno mnożenie, do tego jedno dodawanie i jedno odejmowanie.

8-punktowa FFT

- Dla 8 punktów, dwupunktowe obliczenia wykonuje się w trzech etapach.



Współczynniki *twiddle*

- Współczynniki *W* mają postać:

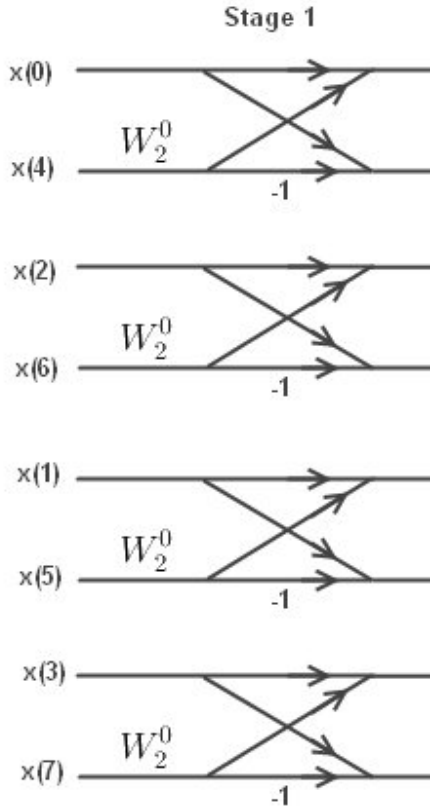
$$W_N^k = e^{-j2\pi k/N} = \cos\left(\frac{2\pi k}{N}\right) - j\sin\left(\frac{2\pi k}{N}\right)$$

- Są to *twiddle factors* („współczynniki skręcenia”).
- *N* jest rozmiarem transformaty na danym etapie.
- Dla *i*-tego etapu potrzeba 2^{i-1} współczynników.
- Współczynniki są stałe dla danego rozmiaru transformaty. Są one zatem obliczane wcześniej i zapisywane w tablicy.

FFT – odwracanie bitów

- Aby ustalić kolejność próbek, stosuje się odwracanie bitów (*bit reversal*) – ustawia się bity „od prawej do lewej”.

0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7



FFT vs DFT

- Porównanie złożoności obliczeniowej:

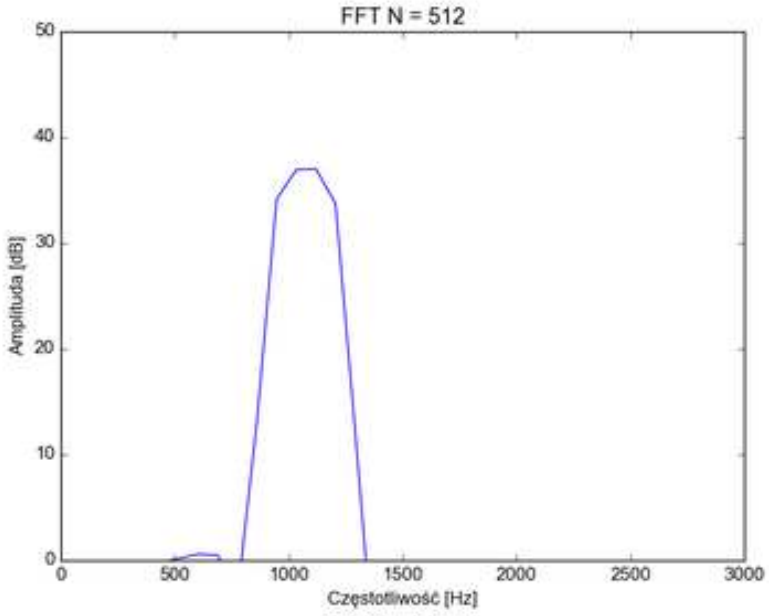
Rozmiar	DFT	FFT
N	N^2	$N \log_2(N)$
512	262 144	4608
1024	1 048 576	10 240
2048	4 194 304	22 528

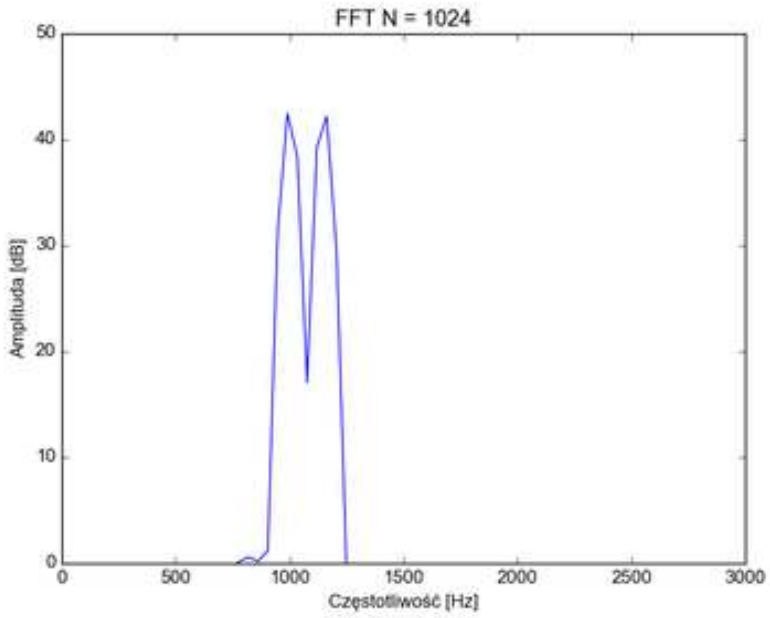
Rozdzielczość częstotliwościowa

- Transformata dla *N* próbek sygnału to *N* punktów zespolonego widma w zakresie od 0 do *fs*.
- Dla sygnału rzeczywistego: dwie kopie widma, *N*/2 punktów w zakresie od 0 do *fs*/2.
- Odstęp między punktami: ***df = fs / N***
  - rozdzielczość częstotliwościowa transformaty.
- Jest to minimalna odległość między składowymi widma, które da się rozróżnić.
- Większy rozmiar *N* – większa (lepsza) rozdzielczość.

Rozdzielczość częstotliwościowa

Praktyczny przykład: y = sin(1000 Hz) + sin(1150 Hz)





N = 512, df = 93,75 Hz

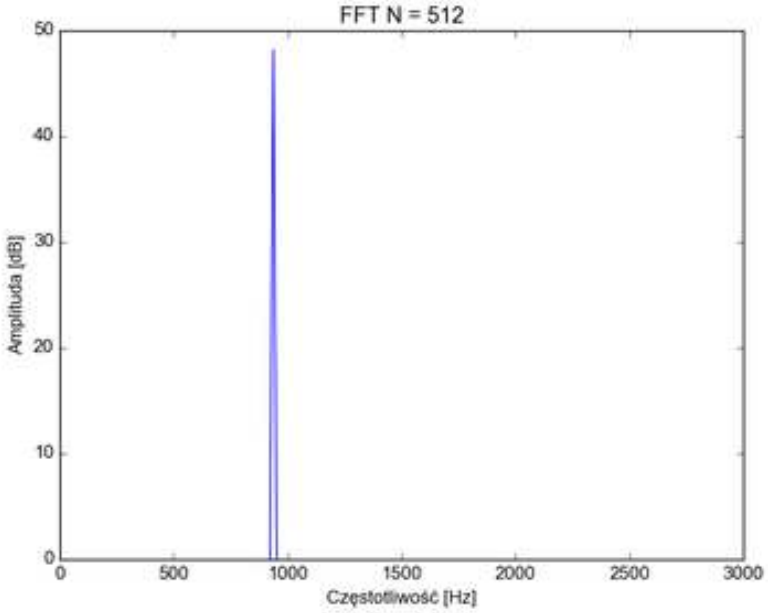
N = 1024, df = 46,88 Hz

Okienkowanie i przecieki widma

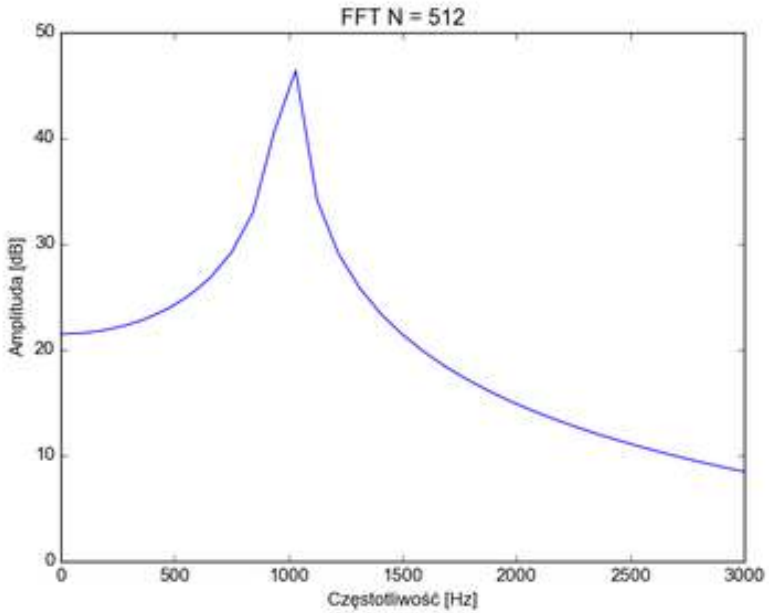
- Transformacja Fouriera zakłada, że przekształcany jest okres sygnału.
- Dla rzeczywistego sygnału: „nieskończony okres”.
- Wycinamy fragment za pomocą okna czasowego.
- Czyli: mnożymy przez funkcję okna.
- W dziedzinie widma: splot z transformatą f. okna.
- Dla okna prostokątnego: transformata = sin(x) / x
- Efekt: przecieki energii prążka widmowego na sąsiednie przedziały częstotliwości.

Okienkowanie i przecieki widma

Efekt przecieków widma dla FFT sinusa (*N* = 512), przy oknie prostokątnym:



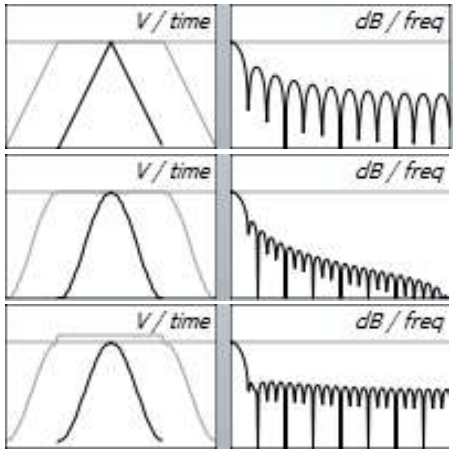
f = 10 * (*fs* / 512) = 937,5 Hz



f = 1000 Hz

Okienkowanie i przecieki widma

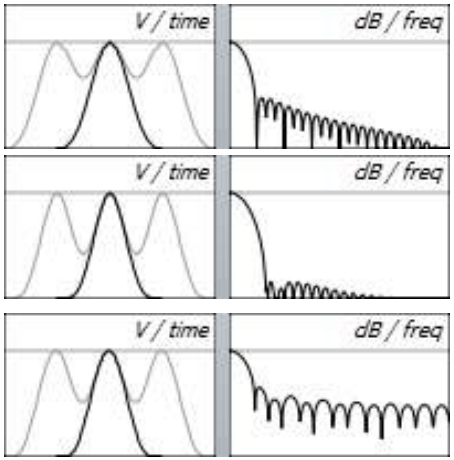
- Funkcje okna zmniejszają przecieki widmowe.
- Sygnał jest mnożony przez funkcję okna przed obliczeniem FFT.



Bartletta

von Hanna

Hamminga



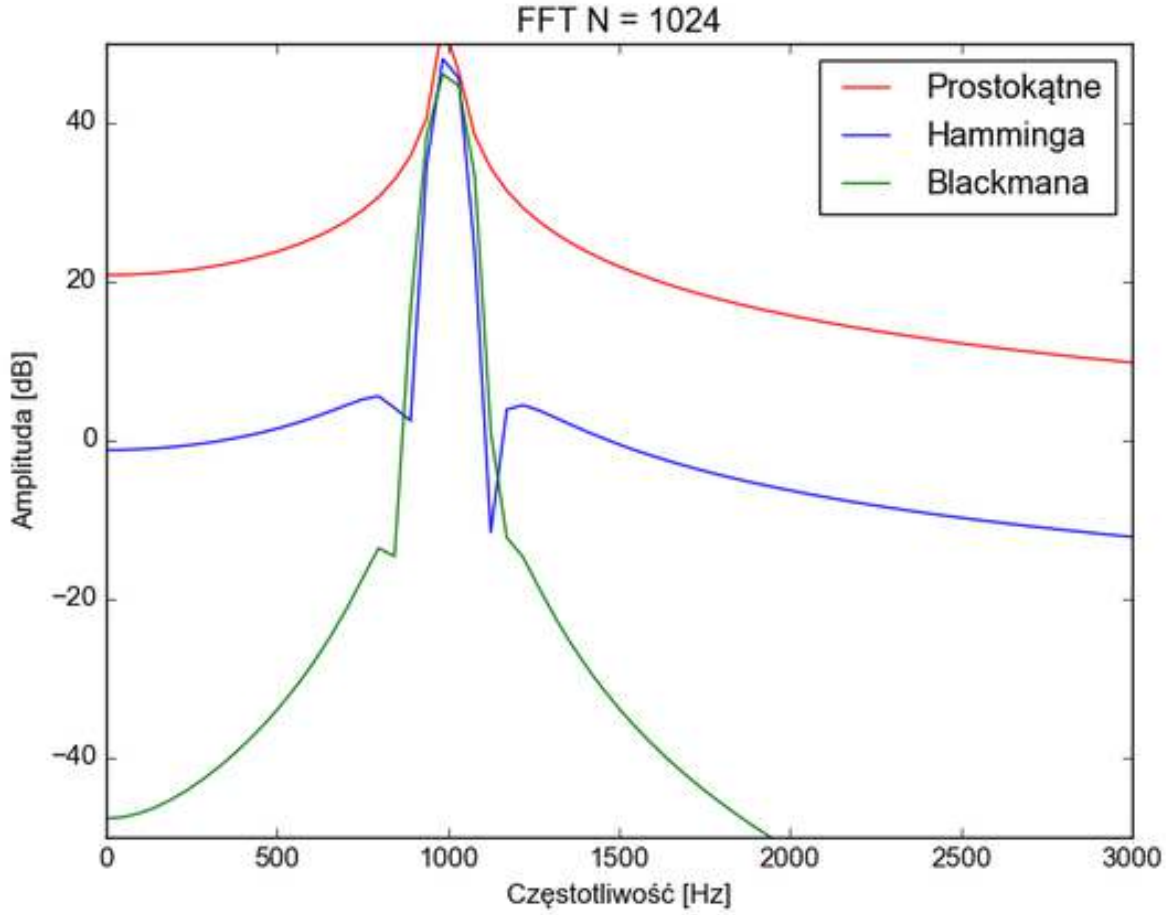
Blackmana

Blackmana-Harrisa

Parzena

Okienkowanie i przecieki widma

Porównanie FFT sinusa dla różnych okien:



Okienkowanie i przecieki widma

Nie ma „najlepszego” okna!

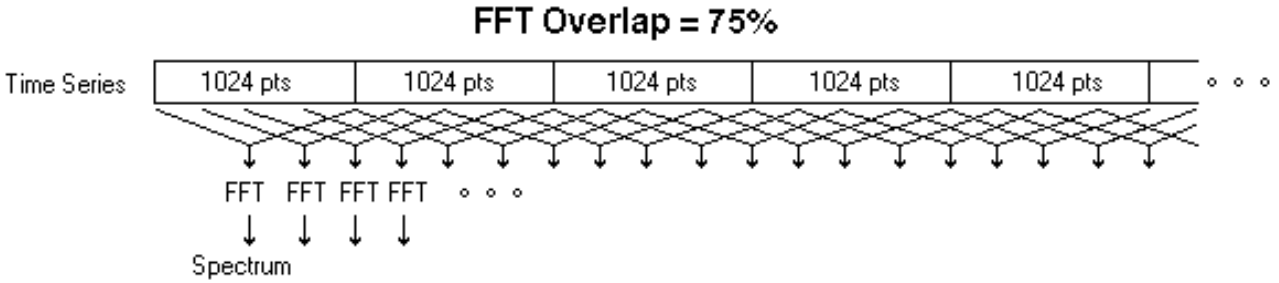
- Okno Hamminga:
  - wąski listek główny – dobra rozdzielczość częstotl.
  - wysoki poziom listków bocznych.
- Okno Blackmana:
  - bardzo dobre tłumienie listków bocznych,
  - szeroki listek główny – gorsza rozdzielczość.
- Musimy wybrać kompromis (jak zawsze).
- W razie wątpliwości: okno Hamminga.

Rozdzielczość czasowa

- Czy to znaczy, że większe okno jest lepsze?
- Rozdzielczość czasowa – zdolność do rozróżnienia dwóch zdarzeń czasowych.
- ***dt = N / fs = 1 / df***
- Dla *fs* = 48 kHz: *N* = 512: *dt* = 10,66 ms
  N = 1024: *dt* = 21,33 ms
- Nie rozróżnimy dwóch zdarzeń w jednym oknie.
- Poprawiając rozdzielczość częstotliwościową, pogarszamy czasową!

STFT i zakładkowanie widma

- Często obliczamy FFT sygnału „blok po bloku” - STFT (*short-time Fourier transform*).
- Typową metodą poprawy rozdzielczości czasowej jest stosowanie zakładki (*overlap*).
- Okno przesuwa się nie o jego rozmiar *N*, a o jego część, zwykle o *N*/2 (50%) lub o *N*/4 (75%).
- Wady: więcej obliczeń, trudna organizacja pamięci.



FFT na DSP

- Nasz DSP (TMS320C5535) ma sprzętową implementację FFT do rozmiaru 1024 włącznie.
- Nie ma sensu pisać własnych implementacji.

Functions	Description
void cfft (DATA *x, ushort nx, type)	Radix-2 complex forward FFT – MACRO
void cfft32 (LDATA *x, ushort nx, type);	32-bit forward complex FFT
void cifft (DATA *x, ushort nx, type)	Radix-2 complex inverse FFT – MACRO
void cifft32 (LDATA *x, ushort nx, type);	32-bit inverse complex FFT
void cbrev (DATA *x, DATA *r, ushort n)	Complex bit-reverse function
void cbrev32 (LDATA *a, LDATA *r, ushort)	32-bit complex bit reverse
void rfft (DATA *x, ushort nx, type)	Radix-2 real forward FFT – MACRO
void rifft (DATA *x, ushort nx, type)	Radix-2 real inverse FFT – MACRO
void rfft32 (LDATA *x, ushort nx, type)	Forward 32-bit Real FFT (in-place)
void rifft32 (LDATA *x, ushort nx, type)	Inverse 32-bit Real FFT (in-place)

Liczby zespolone na DSP

- FFT operuje na liczbach zespolonych.
- Standard zapisu na DSP: część rzeczywista i część zespolona jako osobne liczby, jedna po drugiej: Re(0), Im(0), Re(1), Im(1), Re(2), Im(2), …
- Każda część zapisywana jako Q15 lub Q31.
- Jeżeli funkcja wymaga sygnału zespolonego, a mamy sygnał rzeczywisty, musimy wstawić zera między wartości rzeczywiste.

FFT z DSPLIB

- Dla sygnału rzeczywistego – funkcja *rfft*:

```
void rfft (DATA *x, ushort nx, type);
```

- Rozmiar: do 2048. Funkcja liczy FFT jak z sygnału zespolonego i przekształca wyniki.
- Proste argumenty:
  - *x* – wskaźnik do bufora próbek (zostanie nadpisany przez wynik!),
  - *nx* – rozmiar bufora (liczba próbek),
  - *type* – podajemy *SCALE*.

FFT z DSPLIB

- Funkcje FFT z DSPLIB wymagają spełnienia następujących warunków (dla *N* = 2048).
- Konfiguracja pamięci w pliku *.cmd*:

```
.fftcode        >  SARAM0
.data:twiddle   >  SARAM1, align(2048)
.input          >  DARAM0, align(4)
```

- Deklaracja bufora do obliczania FFT w naszym kodzie

```
#define N 2048
#pragma DATA_SECTION (bufor_fft, ".input")
DATA bufor_fft[N];
```

Zastosowanie okna

Wykorzystanie okna np. Hamminga:

- obliczamy wartości okna dla założonego rozmiaru, za pomocą np. Matlaba lub Pythona,
- zamieniamy na Q15, UWAGA na przepełnienie (częsty błąd): maksymalna wartość to 32767,
- zapisujemy współczynniki w tablicy *const int*,
- mnożymy próbkę sygnału przez odpowiedni współczynnik, najlepiej przed zapisaniem do bufora.

Schemat przetwarzania

FFT wymaga przetwarzania blokowego!

- Odczyt próbki wejściowej.
- Mnożenie przez wartość okna, zapisanie do bufora.
- Jeżeli bufor jest zapełniony:
  - obliczenie FFT,
  - dodatkowe przetwarzanie wg potrzeb,
  - wyniki zapisywane do bufora wyjściowego.
- Wysłanie próbki z bufora wyjściowego na wyjście.

Widmo amplitudowe

Mając FFT, można obliczyć widmo amplitudowe:

$$A = \sqrt{\mathrm{Re}(X)^2 + \mathrm{Im}(X)^2}$$

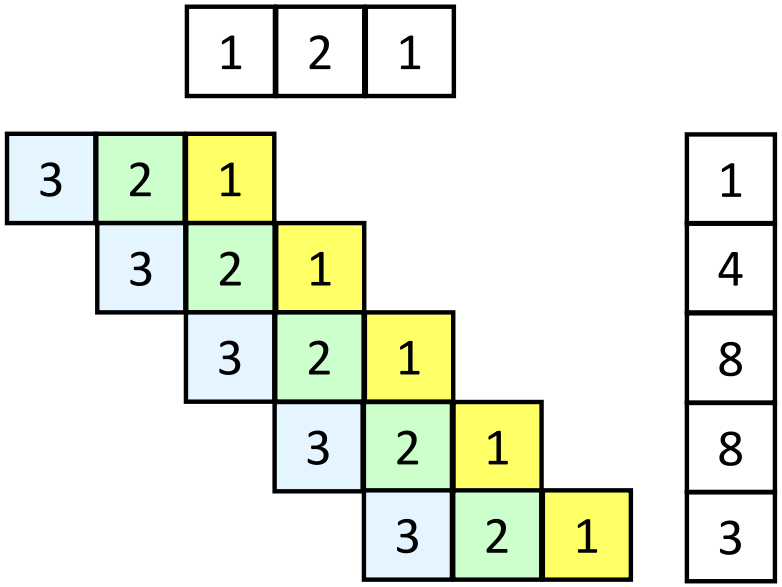
- Przejście pętlą po parach próbek:
  - podniesienie do kwadratu (*_smpy*),
  - dodanie (Re + Im) – UWAGA na przepełnienie,
  - zapisanie sumy – może być w tym samym buforze
- Obliczenie pierwiastka – funkcja *sqrt_16* z DSPLIB, uwaga: z *N*/2 wartości.

Splot

- Splot (*convolution*) jest kolejną z podstawowych operacji przetwarzania sygnałów.
- Z definicji: sumy iloczynów dwóch sygnałów, gdy jeden jest przesunięty względem drugiego, dla wszystkich przesunięć *k* takich, że przynajmniej jedna próbka się pokrywa.
- Wynik splotu sygnałów o długościach *N* i *M* jest sygnałem o długości *N*+*M*-1.
- Funkcje z DSPLIB: *convol* oraz wersje „przyspieszone” *convol1* i *convol2*.

Splot

Przykład: a = [1, 2, 1]; b = [1, 2, 3]



Szybki splot

- Twierdzenie o splocie: splotowi dwóch sygnałów odpowiada mnożenie ich transformat.
- Obliczanie splotu w dziedzinie widma:
  - obliczamy FFT obu sygnałów,
  - mnożymy je,
  - obliczamy IFFT – mamy wynik splotu.
- Jest to „szybki splot” (*fast convolution*) – DSP zawsze obliczy go szybciej, niż w dziedzinie czasu.

Splot w systemach LTI

- LTI (*linear time invariant*) – systemy liniowe o niezmiennej odpowiedzi impulsowej.
- Odpowiedź systemu LTI na sygnał jest równa splotowi sygnału z odpowiedzią impulsową.
- Można wykorzystać szybki splot, przy czym odpowiedź impulsowa jest stała – FFT liczone raz.
- Wiele praktycznych systemów jest traktowanych jako LTI. Można więc wykorzystać szybki splot do wyznaczenia odpowiedzi układu na pobudzenie.

Filtracja blokowa jako splot

- Filtr FIR rzędu 4 (jest LTI). Sygnał x = [1, 2, 3, 4, 5].

h0	h1	h2	h3	h4
1				
2	1			
3	2	1		
4	3	2	1	
5	4	3	2	1

- Wygląda jak splot?
- Filtr FIR można przetwarzać blokowo wykorzystując (szybki) splot. Jest to najszybszy sposób filtracji sygnału na DSP.

Splot ciągłego sygnału

Typowa sytuacja w praktyce:

- jeden sygnał (odpowiedź impulsowa) jest stały, więc przy szybkim splocie wystarczy obliczyć jego FFT tylko raz,
- próbki drugiego sygnału napływają ciągle na wejście (sygnał potencjalnie nieskończony), więc musimy dzielić go na ramki i przetwarzać blok po bloku.

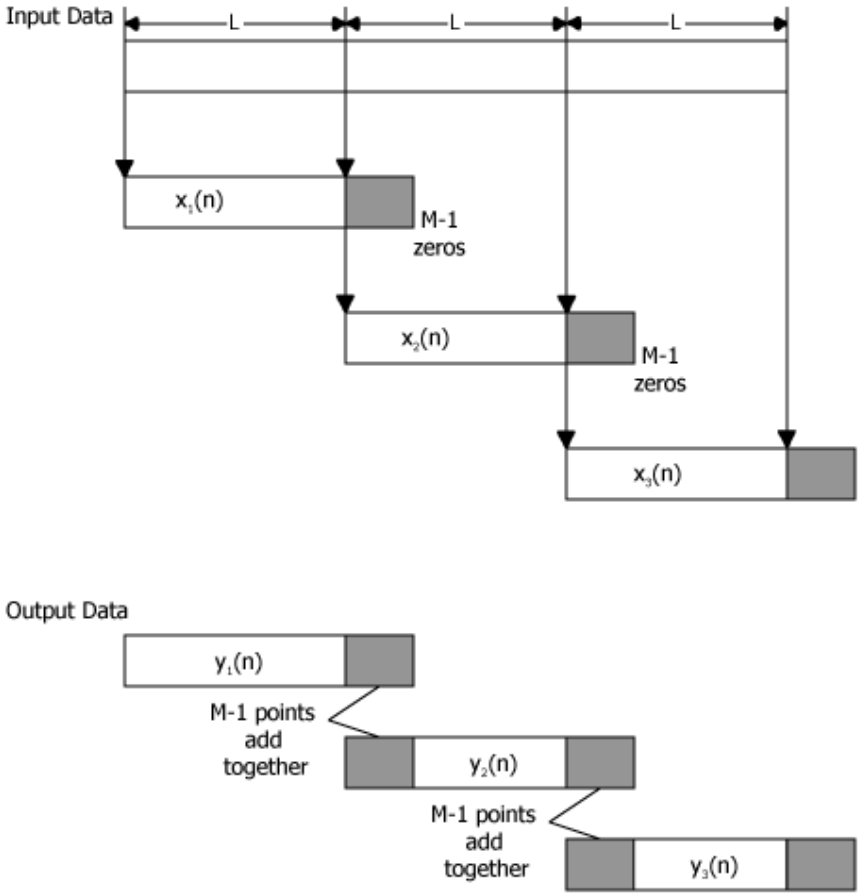
Splot ciągłego sygnału

Powstaje praktyczny problem. Załóżmy, że wykorzystujemy splot do filtracji FIR.

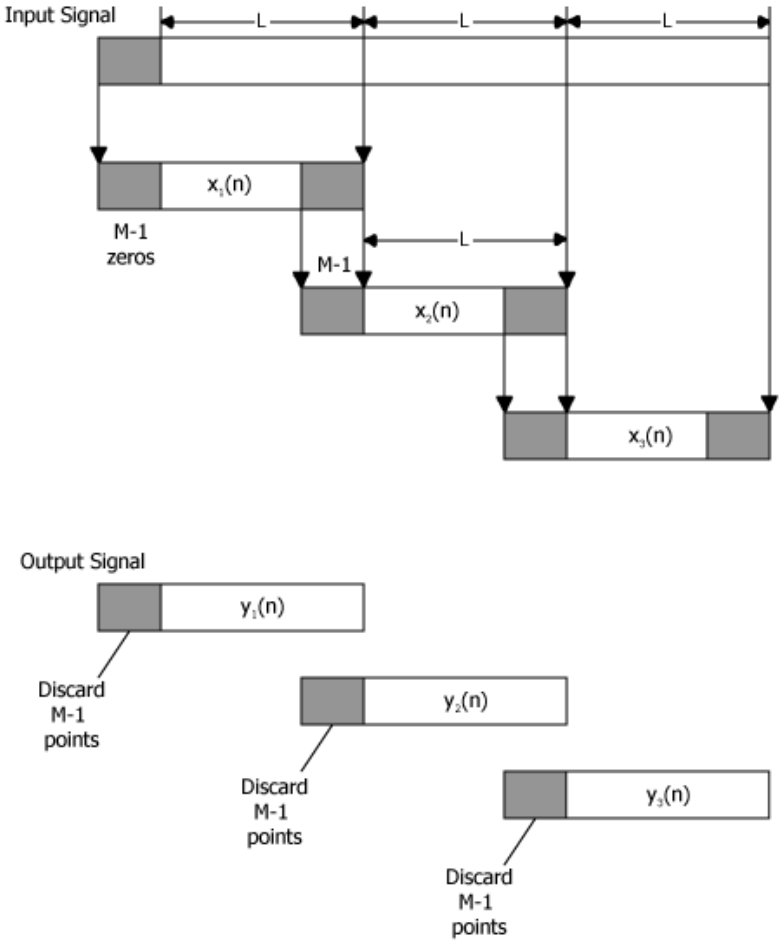
- Aby przefiltrować „najstarsze” próbki, potrzebujemy danych, które były w poprzednim bloku. Jeśli je pominiemy, wynik będzie błędny.
- Z kolei „najnowsze” próbki będą potrzebne w następnym bloku.
- Są dwa najczęściej stosowane algorytmy, które rozwiązują ten problem.

Splot ciągłego sygnału

Overlap-add



Overlap-save



Splot metodą *overlapp-add*

Overlapp-add (OLA):

- sygnał stały, np. odpowiedź impulsowa, długość M,
- sygnał ciągły (np. z wejścia): dzielony na bloki o długości L,
- oba sygnały uzupełniane zerami do dł. $N = M + L - 1$, N powinno być potęgą dwójki,
- obliczenie szybkiego splotu – FFT, mnożenie, IFFT,
- pierwsze ($M - 1$) próbek wyniku jest dodawanych do wyniku z poprzedniego bloku („na zakładkę”).

Splot metodą *overlapp-add*

Przykład praktyczny: filtr FIR, 30 współczynników.

- Ustalamy: $M = 30$, $N = 512 \rightarrow L = 483$.
- Uzupełniamy wsp. filtru 482 zerami, obliczamy FFT.
- Dzielimy sygnał na bloki o dł. $L = 483$ próbki.
  Dla każdego bloku:
  - uzupełniamy 29 zerami,
  - liczymy FFT, mnożymy przez FFT filtru, IFFT,
  - zapisujemy wynik w buforze, nakładając go na ostatnie 29 próbek wyniku i sumując je.

Metoda *overlap-save*

Overlap-save (OLS) – alternatywna metoda:

- sygnał ciągły jest dzielony na bloki o dł. ($L + M - 1$), ale z zakładką na ($M - 1$) próbek,
- szybki splot obu bloków – jak w OLA,
- pierwsze ($M - 1$) próbki wyniku są odrzucane, pozostałe są zapisywane w buforze.

Obie metody (OLA i OLS) mają porównywalną złożoność obliczeniową.

Podsumowanie

Co powinniśmy wiedzieć i umieć?

- zasada działania FFT,
- jaki wpływ ma rozmiar FFT na jej rozdzielczość,
- z czego wynikają przecieki widma i jak je redukować - w jaki sposób używać funkcji okien,
- jak uruchomić FFT i IFFT z DSPLIB,
- jak wykonać szybki splot za pomocą FFT,
- jak użyć FFT do filtracji blokowej FIR,
- jak wykonać splot ciągłego sygnału.