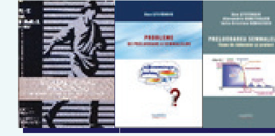
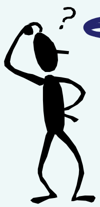
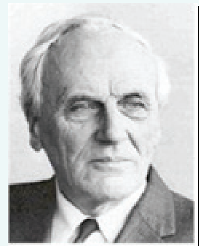


❻ Reprezentări de semnale prin ecuații cu diferențe

Grafuri de semnale (continuare)



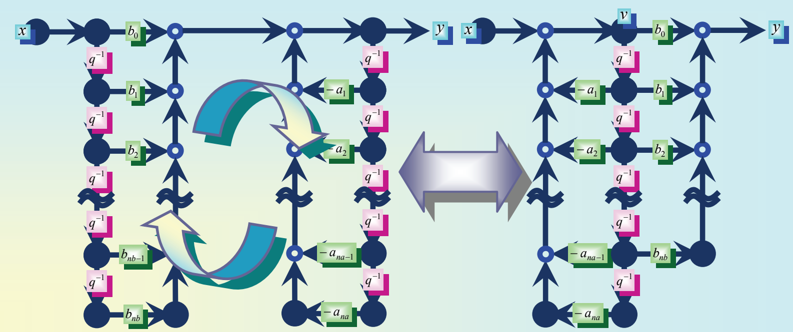
Există o schemă de calcul mai eficientă?



Teorema 3 (Tellegen)

Schema directă de calcul utilizată pentru rezolvarea unei ecuații cu diferențe este echivalentă cu schema obținută prin inversarea blocurilor de intrare şi ieşire, păstrînd semnalele de intrare şi ieşire în pozițiile inițiale.

Demonstrație (schiță)



Ecuația cu diferențe se poate exprima în mod echivalent cu ajutorul polinoamelor Laurent:

$$\boxed{y[n]=\frac{B\left(q^{-1}\right)}{A\left(q^{-1}\right)}x[n]} \quad \boxed{\forall\, n\in\mathbb{Z}}\,.$$

Se poate evalua mai întîi ieşirea cauzală a filtrului IIR care are numai poli:

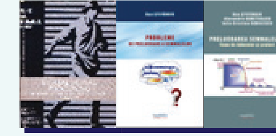
$$v[n]\stackrel{def}{=}\frac{1}{A\left(q^{-1}\right)}x[n],\ \ \forall n\geq 0 \qquad \Leftrightarrow \qquad v[n]=x[n]-a_1v[n-1]-\cdots-a_{na}v[n-na],\ \ \forall n\geq 0\ .$$

Apoi, se poate evalua ieşirea cauzală a filtrului FIR stimulat cu ieşirea filtrului IIR:

$$y[n]=B\left(q^{-1}\right)v[n],\ \ \forall n\geq 0 \qquad \Leftrightarrow \qquad y[n]=b_0v[n]+b_1v[n-1]+\cdots+b_{nb}v[n-nb],\ \ \forall n\geq 0\ .$$

☝ **Este exploatată proprietatea de comutativitate a SLID înseriate!**

❻ Reprezentări de semnale prin ecuaţii cu diferenţe



Grafuri de semnale (continuare)

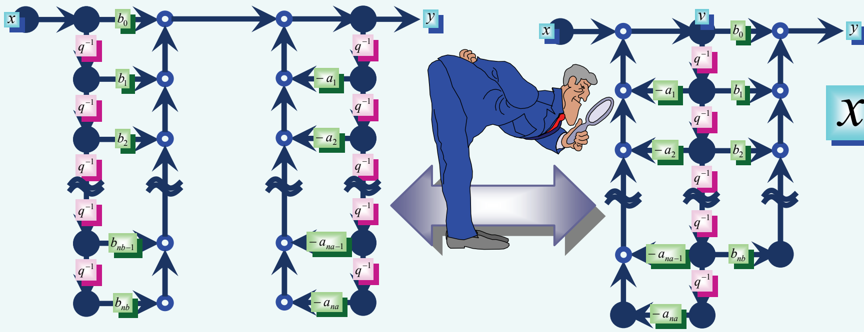
Schema eficientă de calcul recursiv (forma directă II)

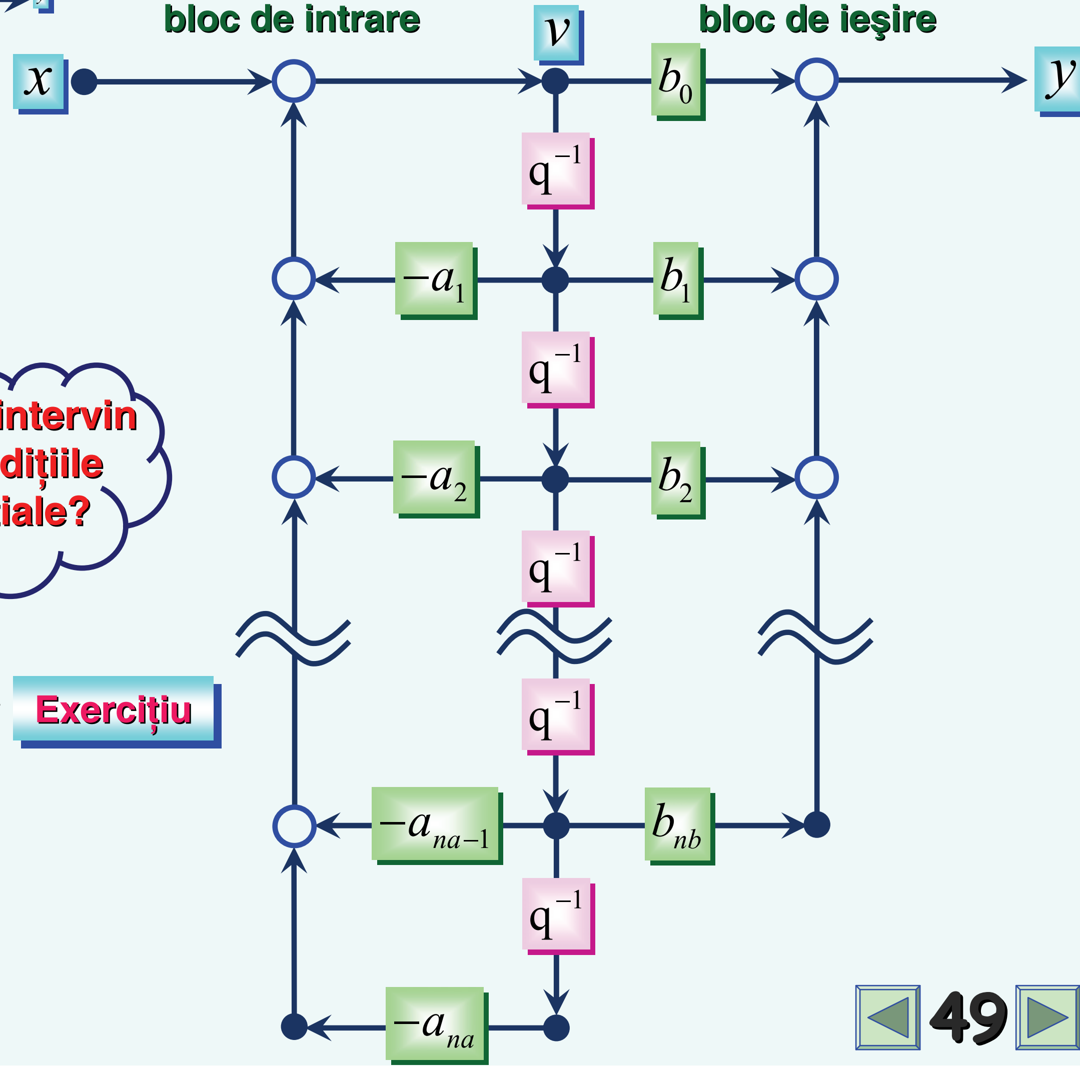
$$v[n] \overset{def}{=} \frac{1}{A(q^{-1})} x[n]$$

bloc de intrare

$$y[n] = B(q^{-1}) v[n]$$

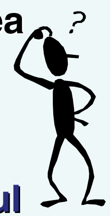
bloc de ieşire



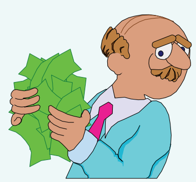


Avantajele implementării

- Blocuri de întîrziere **ne-redundante**.
- Pentru a calcula valoarea ieşirii la un anumit moment normalizat, **numai blocul de intrare trebuie utilizat în calculul valorilor anterioare, blocul de ieşire fiind utilizat o singură dată, la sfîrşit.**

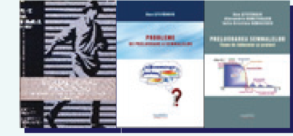


Cum intervin condiţiile iniţiale?



Exerciţiu

❻ Reprezentări de semnale prin ecuații cu diferențe



Grafuri de semnale (continuare)

Schema alternativă de calcul recursiv (forma transpusă II)

Există o schemă de calcul și mai eficientă?

Oarecum. Se poate reduce și numărul de sumatoare.

$$y[n] = -\sum_{p=1}^{na} a_p y[n-p] + \sum_{m=0}^{nb} b_m x[n-m]$$

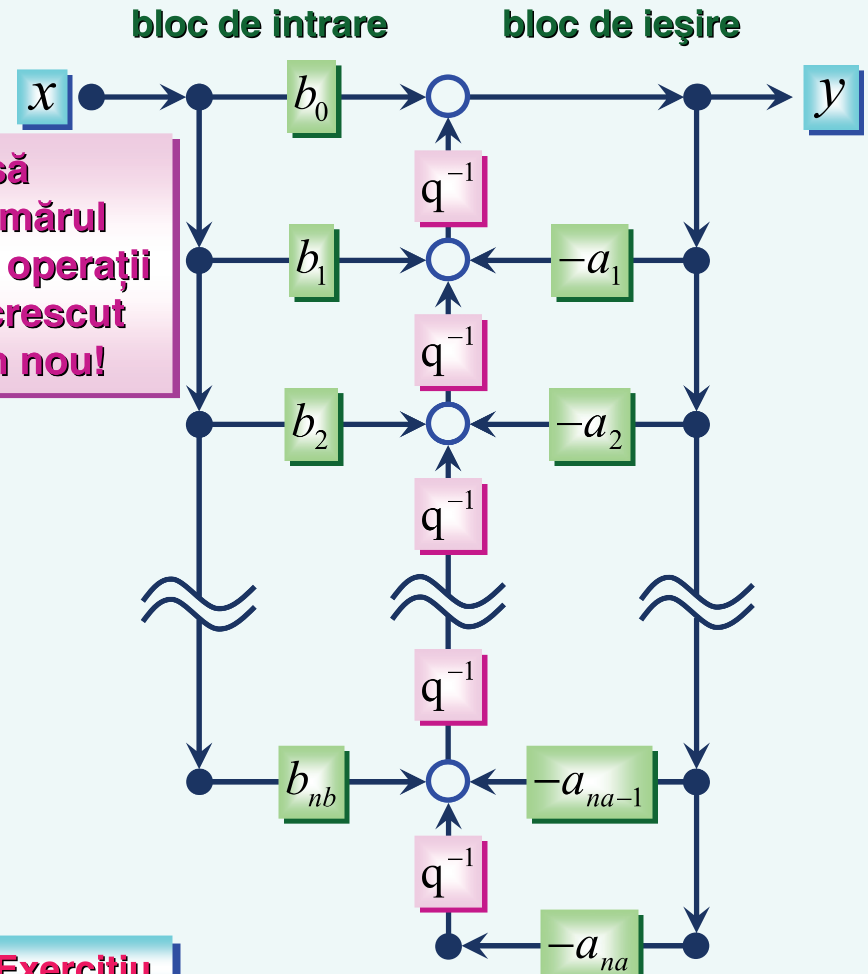
$\forall n \in \mathbb{N}$

$$\Updownarrow$$

$$\begin{aligned} y[n] = {} & b_0 x[n] + \\ & + q^{-1}\left(b_1 x - a_1 y\right)[n] + \\ & + q^{-2}\left(b_2 x - a_2 y\right)[n] + \\ & \dots\dots\dots\dots\dots\dots \\ & + q^{-k}\left(b_k x - a_k y\right)[n] + \\ & \dots\dots\dots\dots\dots\dots \end{aligned}$$

$\forall n \in \mathbb{N}$

☝ Observați gruparea termenilor în funcție de întîrzieri!

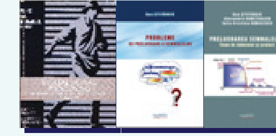


☝ Însă numărul de operații a crescut din nou!

Exercițiu

- Arătați cum se poate implementa schema de mai sus, astfel încît să se evite bucla algebrică.

❻ Reprezentări de semnale prin ecuații cu diferențe



Grafuri de semnale (final)

Condiții inițiale:

$$y[0] = y_0$$

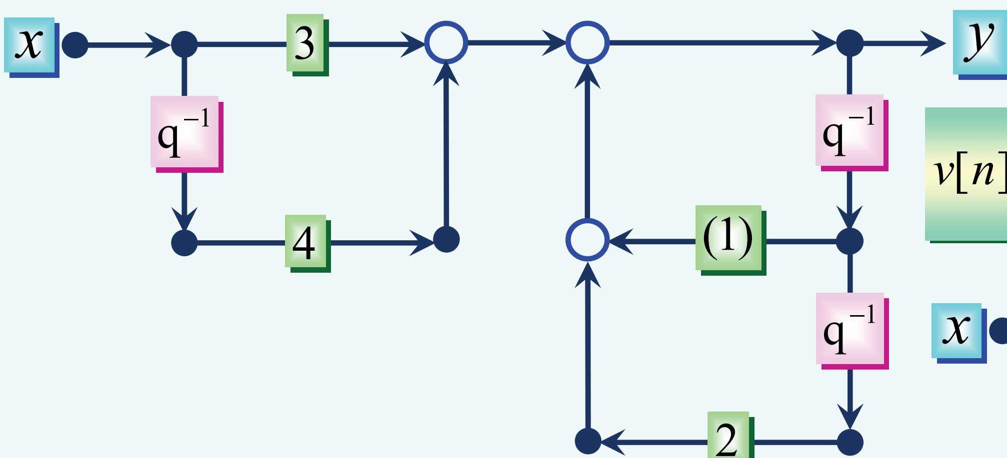
$$y[-1] = y_{-1}$$

Exemplu: o ecuație de ordin II

$$y[n] - y[n-1] - 2y[n-2] = 3x[n] + 4x[n-1] \quad \forall n \in \mathbb{Z}$$

Schema directă I

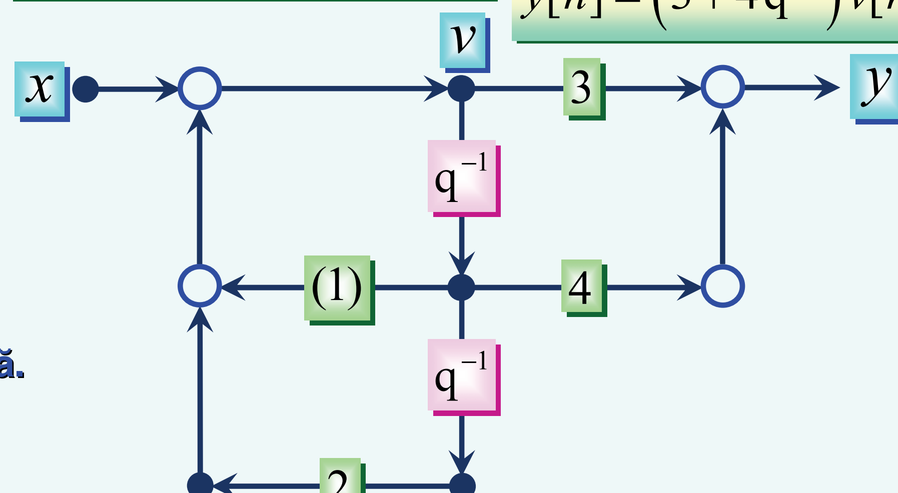
$$3x[n] + 4x[n-1] \qquad y[n-1] + 2y[n-2]$$



Schema directă II

$$v[n] \overset{def}{=} \frac{1}{1 - q^{-1} - 2q^{-2}}\, x[n]$$

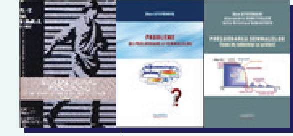
$$y[n] = \left(3 + 4q^{-1}\right) v[n]$$



Exerciții

- ➔ Trasați schema în forma transpusă II aferentă.
- ➔ Calculați $y[3]$ cu toate cele 3 scheme și comparați efortul de calcul.
- ➔ Calculați $y[-4]$ cu schemele aferente (anti-cauzale) și comparați efortul de calcul.

❼ Algoritmi fundamentali de tip FFT



❼.❶ Obiectiv

$$TFD_N(x)[k] \overset{def}{=} X[k] \overset{def}{=} \sum_{n=0}^{N-1} x[n]\, w_N^{nk}$$

$$\forall\, k \in \overline{0, N-1}$$

Implementarea eficientă a TFD

☝ **Relația de calcul pe care se fundamentează procesoarele de semnal.**

Efortul de calcul al definiției.

$$X[k] = \sum_{n=0}^{N-1} \underbrace{\left[\mathrm{Re}(x[n])\cos\omega_{nk}^{N} + \mathrm{Im}(x[n])\sin\omega_{nk}^{N}\right]}_{\text{două înmulțiri și o adunare}} + j \sum_{n=0}^{N-1} \underbrace{\left[\mathrm{Im}(x[n])\cos\omega_{nk}^{N} - \mathrm{Re}(x[n])\sin\omega_{nk}^{N}\right]}_{\text{două înmulțiri și o adunare}}$$

$$\forall\, k \in \overline{0, N-1}$$

2N înmulțiri și 2N-1 adunări | **2N înmulțiri și 2N-1 adunări**

$$\mathcal{O}_{\mathrm{TFD}}[N] = \left[4N^2\right]_{\bullet} + \left[2N(2N-1)\right]_{+} \sim 4N^2$$

Prin exploatarea proprietăților armonicelor elementare.

- **O uşoară scădere a efortului de calcul se obține prin contorizarea doar a operațiilor cu operanzi nebanali (adică pentru $n \neq 0$ și $k \neq 0$).**

$$\mathcal{O}_{\mathrm{TFD}}[N] = \left[4(N-1)^2\right]_{\bullet} + \left[2(N-1)(2N-1)\right]_{+} \sim 4(N-1)^2$$

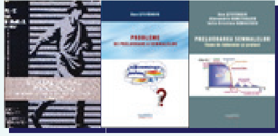
Exercițiu

- **O reducere de aproape 2 ori a efortului de calcul se obține pe seama simetriei/periodicității armonicelor elementare.**

$$w_N^{k(N-n)} = \overline{w_N^{kn}}$$

$$\forall\, n,k \in \overline{0, N-1}$$

❼ Algoritmi fundamentali de tip FFT



❼.❶ Obiectiv

- Un algoritm care conduce la o **reducere de 4 ori** a efortului de calcul este bazat pe **Metoda lui (Gerald) Goertzel**.
- Acest algoritm exploatează periodicitatea armonicelor elementare.

☝ **Pentru a obține o reducere semnificativă a efortului de calcul, proprietățile armonicelor elementare trebuie combinate cu o aranjare eficientă a datelor.**

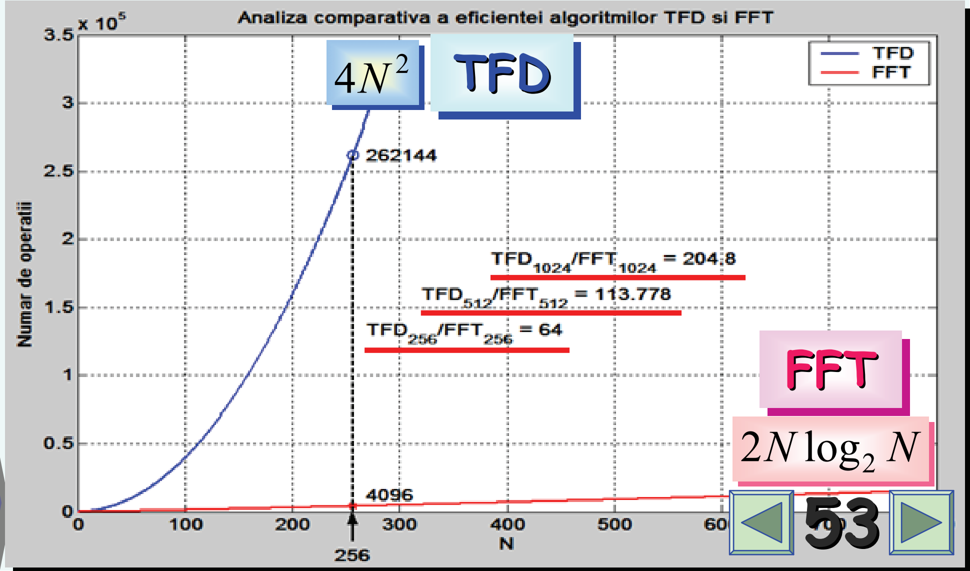
Algoritmi Fourier rapizi
(de tip FFT – *Fast Fourier Transform*)

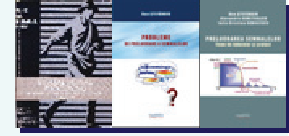
Algoritmi bazați pe segmentarea în timp	Algoritmi bazați pe segmentarea în frecvență	Algoritmi cu structură internă variabilă	Algoritmi de tip Singleton (cu structură internă constantă)

Algoritmi Cooley-Tukey **(1965)**

Cu ajutorul algoritmilor din familia FFT, se obține un efort de calcul **sensibil mai mic**.

Analiză comparativă a eficienței algoritmilor TFD (definiție) și FFT





❼.❷ Algoritmul lui Goertzel

Prima variantă de calcul a TFD

Exprimare echivalentă a TFD

$$X[k] = \sum_{n=0}^{N-1} x[n]\overline{w}_N^{k(N-n)}, \quad \forall k \in \overline{0, N-1}$$

$$\left(w_{\cancel{N}}^{k\cancel{N}} = 1 \quad \forall k \in \overline{0, N-1} \right)$$

periodicitate

sumă de convoluție

ieşirea la momentul *N* a unui SLID

$$x \longrightarrow \boxed{*h_k} \longrightarrow y_k \equiv x * h_k$$

$$h_k[p] \overset{def}{=} \overline{w}_N^{kp} u_0[p], \ \forall p \in \mathbb{Z}$$

$$\mathrm{Supp}(x) = \overline{0, N-1}$$

$$y_k[p] = \sum_{n\geq 0} x[n]h_k[p-n] = \sum_{n=0}^{N-1} x[n]\overline{w}_N^{k(p-n)} \quad \forall p \in \overline{0, N-1}$$

$$X[k] = y_k[N], \quad \forall k \in \overline{0, N-1}$$

Funcția de transfer a sistemului

$$H_k(z) \overset{def}{=} \sum_{p\geq 0} h_k[p]z^{-p} = \sum_{p\geq 0} \left(\overline{w}_N^k z^{-1}\right)^p = \frac{1}{1-\overline{w}_N^k z^{-1}}, \quad \forall k \in \overline{0, N-1} \quad |z| > 1$$

Teorema întîrzierii

$$z^{-1}\mathscr{Z}(f)(z) = \mathscr{Z}(q^{-1}f)(z)$$

Ecuația recursivă a ieşirii

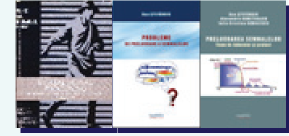
$$y_k[n] - \overline{w}_N^k y_k[n-1] = x[n] \quad \forall n \in \mathbb{Z}$$

Ecuația recursivă a TFD

$$\begin{cases} y_k[0] = x[0] \\ y_k[1] = \overline{w}_N^k y_k[0] + x[1] \\ \vdots \\ y_k[N] = \overline{w}_N^k y_k[N-1] + x[N] = \overline{w}_N^k y_k[N-1] = X[k] \end{cases} \quad \forall k \in \overline{0, N-1}$$

❼ Algoritmi fundamentali de tip FFT

❼.❷ Algoritmul lui Goertzel



A doua variantă de calcul a TFD (îmbunătățită)

Exprimare echivalentă ecuației recursive anterioare

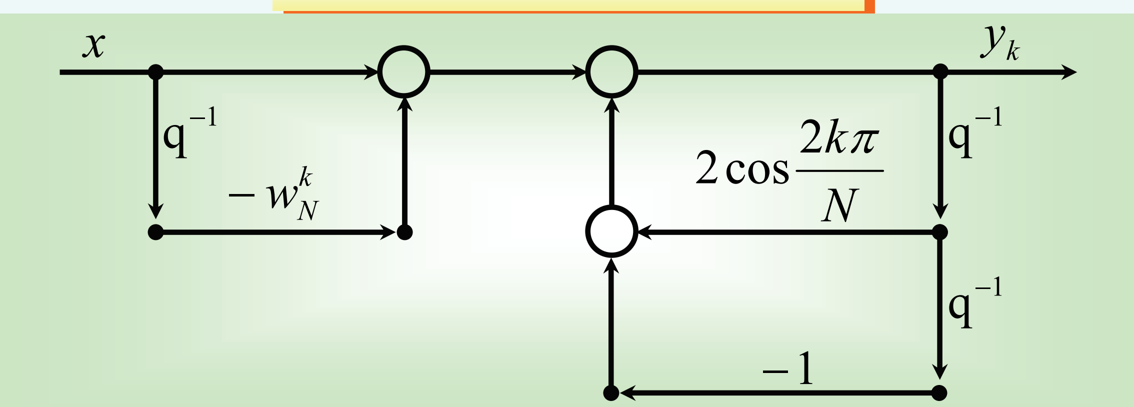
$$\left(1-\overline{w}_N^k \mathrm{q}^{-1}\right) y_k[n] = x[n], \quad \forall n \in \overline{0,N}, \ \forall k \in \overline{0,N-1}$$

înmulțire forțată cu $\left(1-w_N^k \mathrm{q}^{-1}\right)$

$$\left(1-2\cos\frac{2k\pi}{N}\mathrm{q}^{-1}+\mathrm{q}^{-2}\right) y_k[n] = \left(1-w_N^k \mathrm{q}^{-1}\right) x[n], \quad \forall n \in \overline{0,N}, \ \forall k \in \overline{0,N-1}$$

$$y_k[n] = 2y_k[n-1]\cos\frac{2k\pi}{N} - y_k[n-2] + x[n] - w_N^k x[n-1], \quad \forall n \in \overline{0,N}, \ \forall k \in \overline{0,N-1}$$

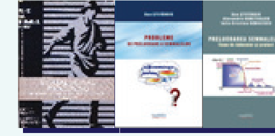
Schema de calcul directă I



Inițializare

$$y_k[-1] = y_k[-2] = 0$$

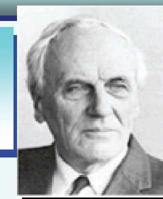
$$\forall k \in \overline{0,N-1}$$



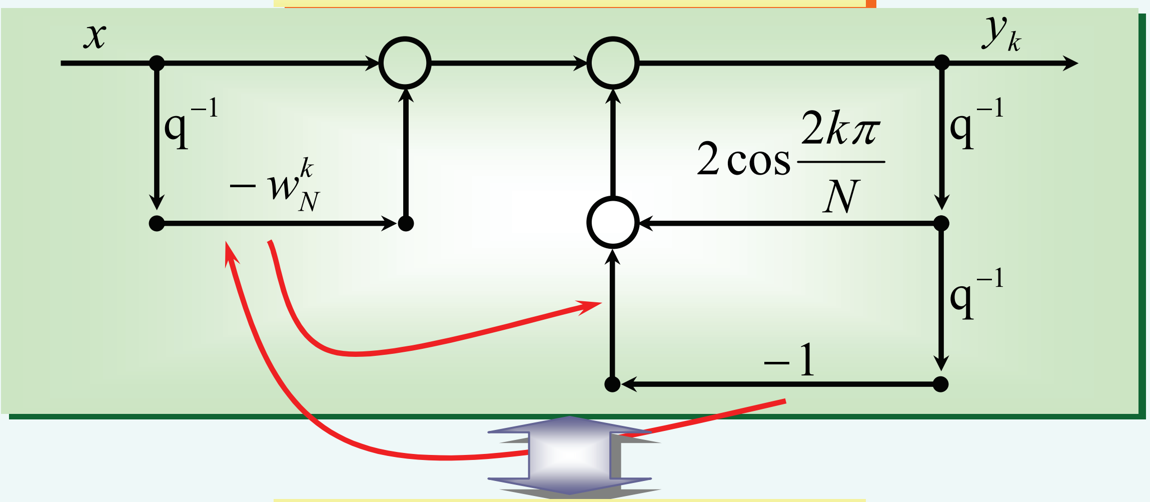
❼.❷ Algoritmul lui Goertzel

Varianta eficientă de calcul a TFD

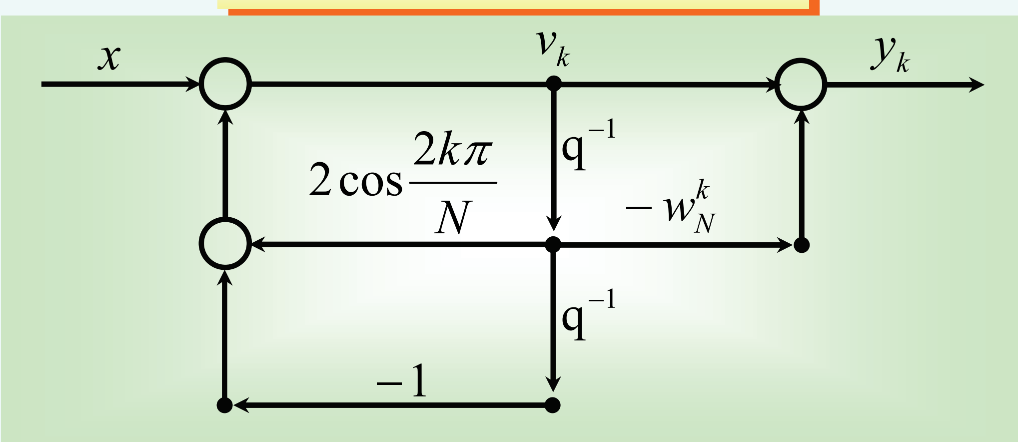
Schema anterioară de calcul se poate transforma echivalent, folosind Teorema lui TELLEGEN.



Schema de calcul directă I



Schema de calcul directă II



Algoritmul lui Goertzel

$$\begin{cases} v_k[-2] = v_k[-1] = 0 \\ \vdots \\ v_k[n] = 2\cos\frac{2k\pi}{N} v_k[n-1] - v_k[n-2] + x[n] \\ \vdots \\ v_k[N] = 2\cos\frac{2k\pi}{N} v_k[N-1] - v_k[N-2] \end{cases}$$

$$X[k] = y_k[N] = v_k[N] - w_N^k v_k[N-1]$$

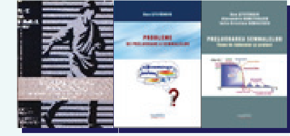
$$\forall k \in \overline{0, N-1}$$

Număr de operații

$$\mathcal{O}_1[N] = \left[\left\lceil \frac{N-1}{2} \right\rceil (2N+5)\right]_{\bullet} + [4N(N+1)]_{+} \sim N^2$$

☝ de 4 ori mai mic

❼ Algoritmi fundamentali de tip FFT



❼.❸ Algoritmul bazat pe segmentarea în timp (FFT-t)

Principiul segmentării în timp

TFD

$$X[k] \overset{def}{=} \sum_{n=0}^{N-1} x[n] w_N^{nk}, \quad \forall k \in \overline{0, N-1}$$

Exprimare echivalentă a TFD, pentru $N = 2M$

$$X[k] = \underbrace{\sum_{m=0}^{M-1} x[2m] w_{2M}^{2mk}}_{\text{Segment cu eşantioane de ordin par}} + \underbrace{\sum_{m=0}^{M-1} x[2m+1] w_{2M}^{(2m+1)k}}_{\text{Segment cu eşantioane de ordin impar}} \quad \forall k \in \overline{0, N-1}$$

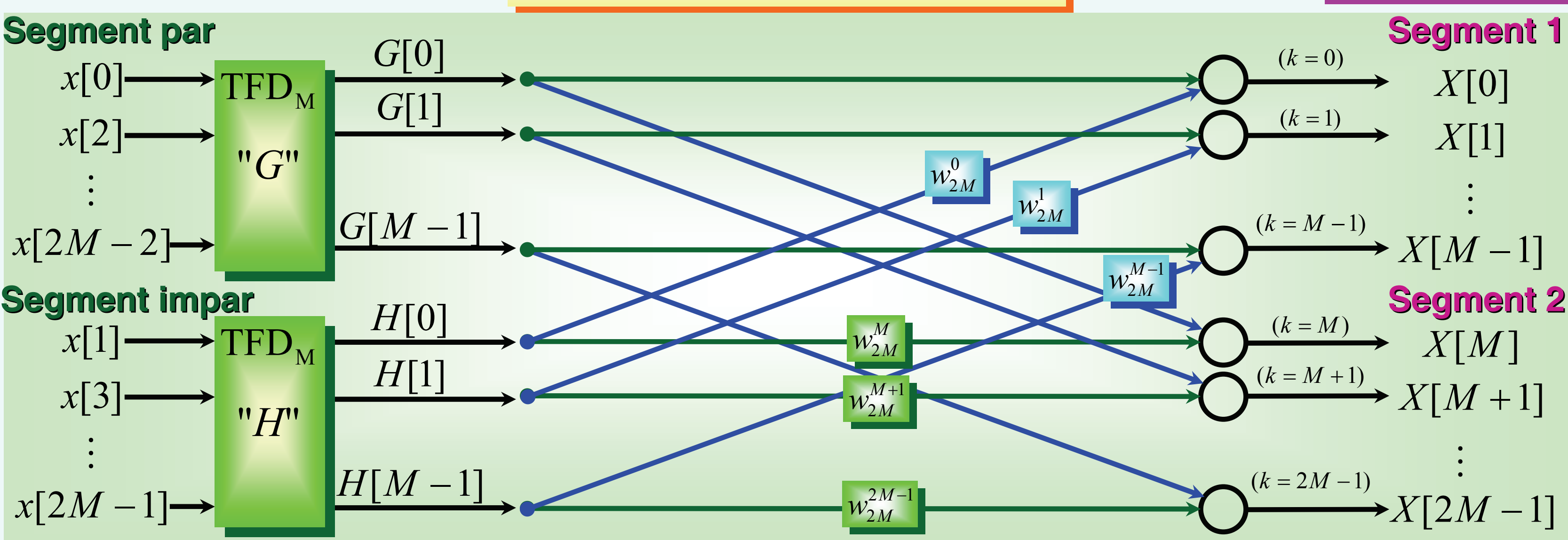
$$w_{\cancel{2}M}^{\cancel{2}mk} = w_M^{mk}$$

$$X[k] = \underbrace{\sum_{m=0}^{M-1} x[2m] w_M^{mk}}_{G[k]} + w_{2M}^{k} \underbrace{\sum_{m=0}^{M-1} x[2m+1] w_M^{mk}}_{H[k]} \quad \forall k \in \overline{0, N-1}$$

☝ **În calculul TFD_{2M} se poate utiliza o pereche de TFD_M aplicate segmentelor par şi impar ale semnalului original.**

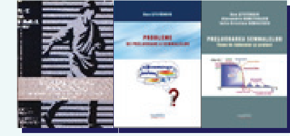
Schema globală de calcul

☝ **Periodicitatea lui w_M^{mk}**



❼ Algoritmi fundamentali de tip FFT

❼.❸ Algoritmul bazat pe segmentarea în timp (FFT-t)



Dacă $N = 4M$, semnalul poate fi partajat în 4 segmente (fiecare segment din perechea anterioară este împărțit într-o nouă pereche de segmente, mai scurte).

TFD_{4M} poate fi evaluată cu ajutorul a 4 TFD_M

În general, dacă $N = 2^L$, semnalul poate fi partajat în 2^{L-1} segmente (fiecare segment avînd doar 2 eşantioane).

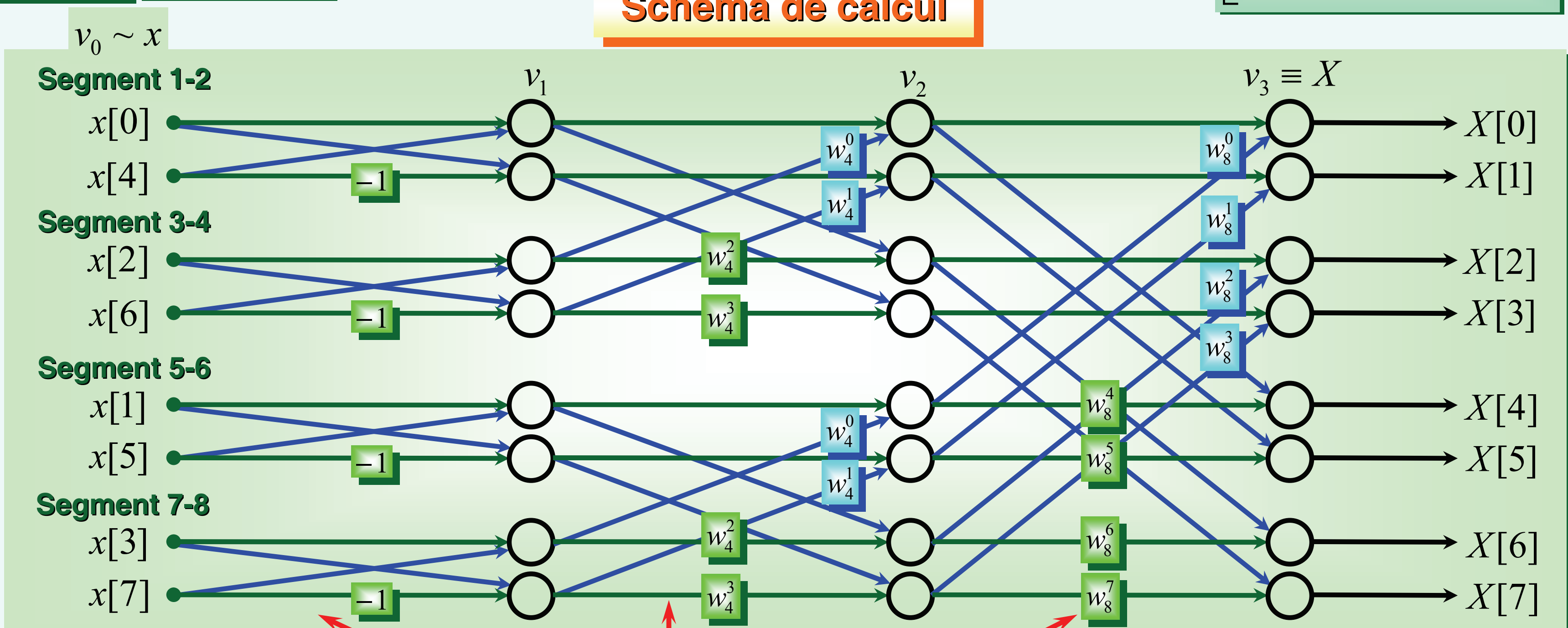
TFD_N poate fi evaluată cu ajutorul a $(L-1)$ TFD_2

$$\begin{cases} X[0] = x[0] + x[1] \\ X[1] = x[0] - x[1] \end{cases}$$

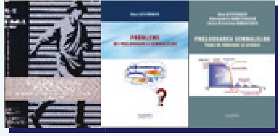
☝ **Tehnica *divide and conquer*.**

Exemplu $N = 8 = 2^3$

Schema de calcul



☝ **3 trepte de calcul paralel**

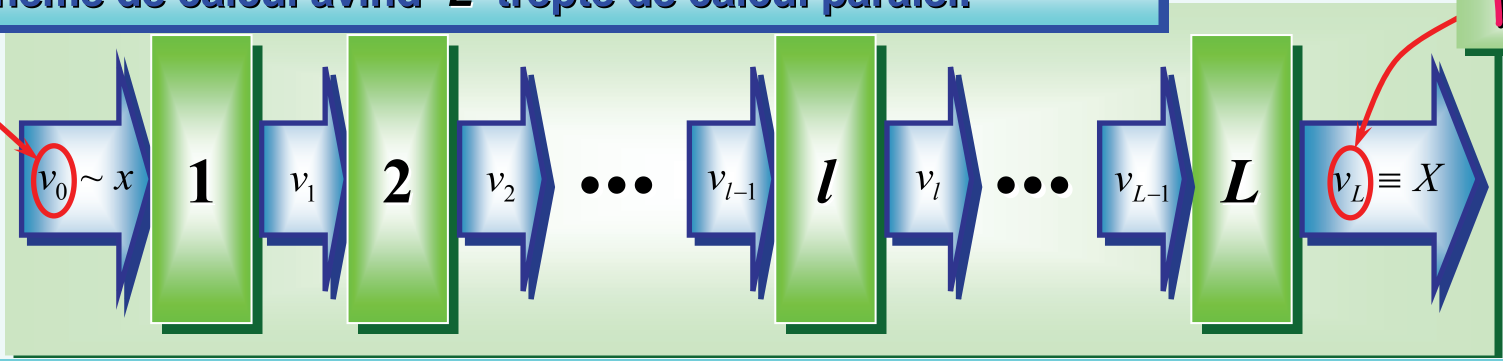


❼.❸ Algoritmul bazat pe segmentarea în timp (FFT-t)

Prima variantă de calcul

În general, dacă $N = 2^L$, valorile TFD_N se pot obține cu ajutorul unei scheme de calcul avînd L trepte de calcul paralel.

versiunea grosieră a TFD_N



TFD_N

Semnalul inițial, v_0, obținut prin rearanjarea eşantioanelor lui x, este transformat succesiv în semnalele intermediare v_1, v_2, ..., v_{L-1}, semnalul final v_L fiind identic cu TFD_N.

☝ Fiecare semnal are $N = 2^L$ eşantioane.

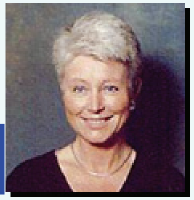
$$v_l \equiv \left[v_l[1] \;\; v_l[2] \;\; \cdots \;\; v_l[N] \right]^T, \;\; \forall l \in \overline{0,L}$$

Aranjarea eşantioanelor lui X în v_L :

$$v_L[k] = X[k-1], \;\; \forall k \in \overline{1,2^L}$$

☜ Nu este necesară rearanjarea eşantioanelor.

... şi în anii '90



Practic, semnalul inițial este rafinat succesiv pînă cînd se obține rezultatul final.

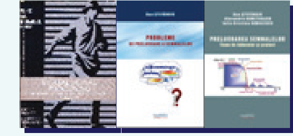
Proces similar developării hîrtiei fotografice.

☝ Una dintre cele mai utilizate imagini în compresia de date.

Fotomodelul suedez Lena Soderberg în anii '70

❼ Algoritmi fundamentali de tip FFT

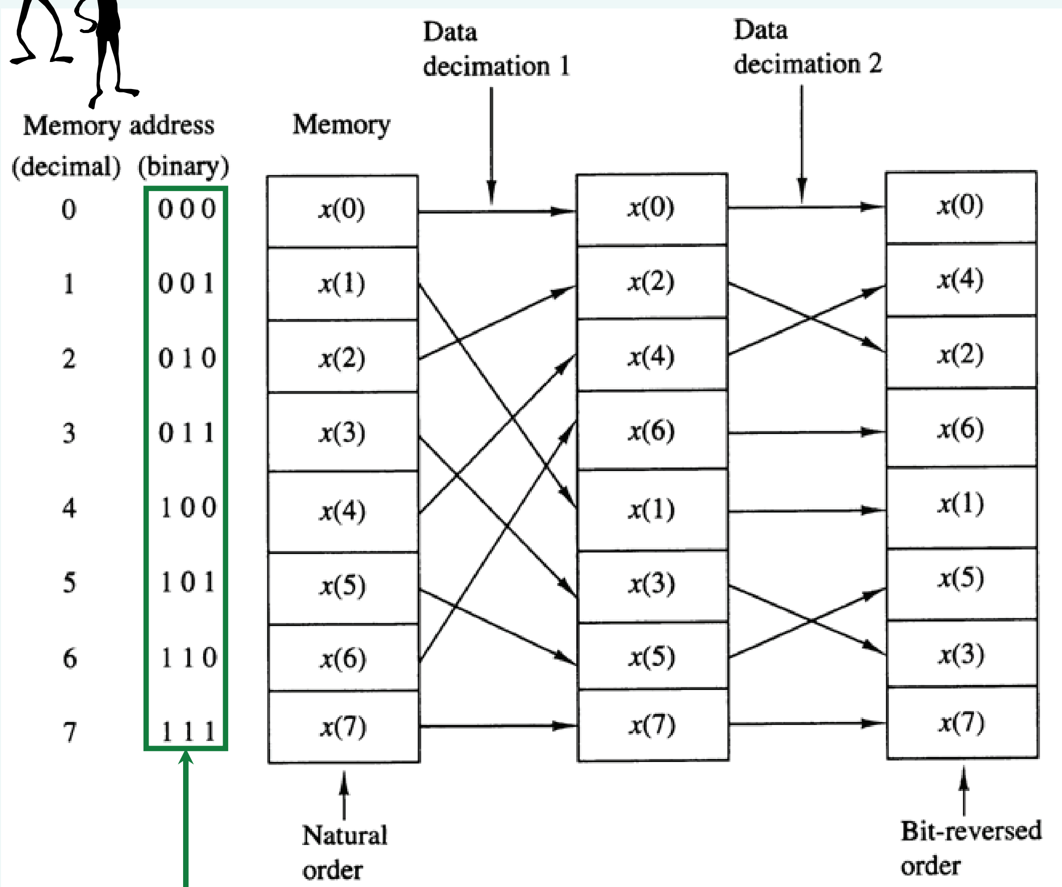
❼.❸ Algoritmul bazat pe segmentarea în timp (FFT-t)



Prima variantă de calcul (continuare)

Cum se poate realiza aranjarea eşantioanelor lui x în v_0?

Prin inversarea binară a indexului inițial.

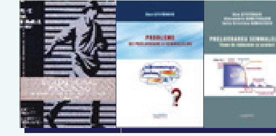


☝ **Observați şi maniera de indexare pe nivelul intermediar.**

$(n_2 n_1 n_0)$	$\rightarrow$	$(n_0 n_2 n_1)$	$\rightarrow$	$(n_0 n_1 n_2)$
(0 0 0)	→	(0 0 0)	→	(0 0 0)
(0 0 1)	→	(1 0 0)	→	(1 0 0)
(0 1 0)	→	(0 0 1)	→	(0 1 0)
(0 1 1)	→	(1 0 1)	→	(1 1 0)
(1 0 0)	→	(0 1 0)	→	(0 0 1)
(1 0 1)	→	(1 1 0)	→	(1 0 1)
(1 1 0)	→	(0 1 1)	→	(0 1 1)
(1 1 1)	→	(1 1 1)	→	(1 1 1)

Reprezentare binară a indexului inițial (pe L biți)

Reprezentare binară a indexului de segment (tot pe L biți)



❼.❸ Algoritmul bazat pe segmentarea în timp (FFT-t)

Prima variantă de calcul (final)

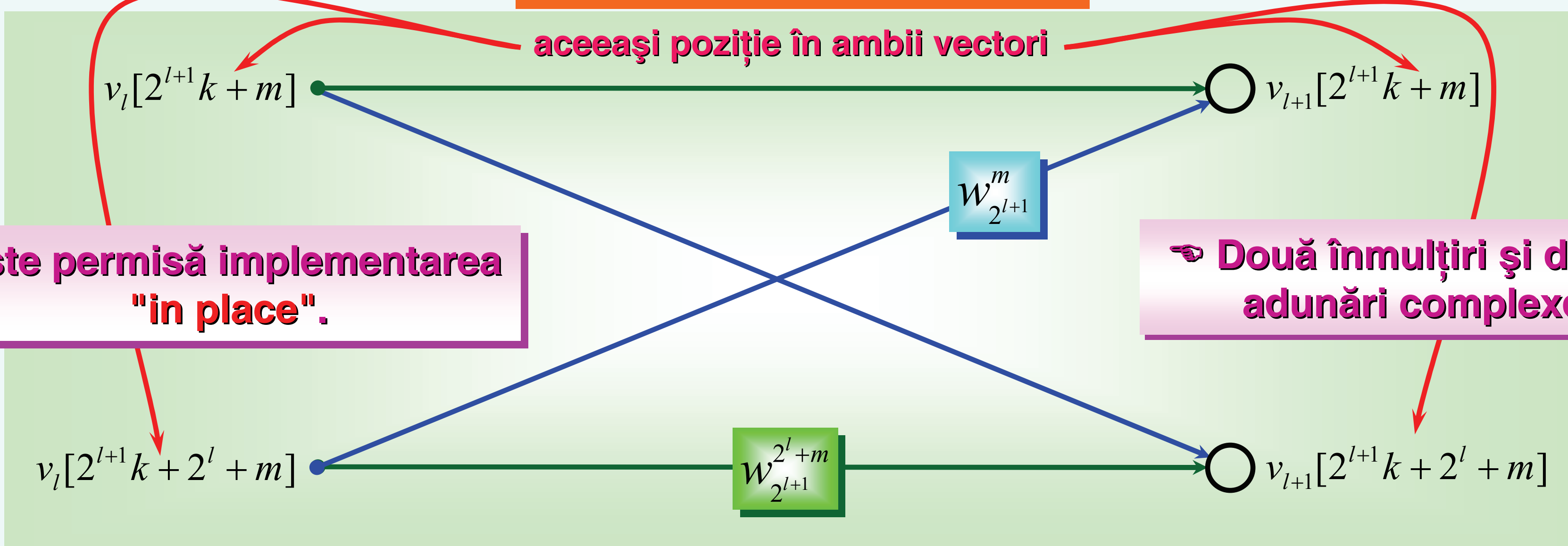
☝ Prima treaptă de calcul este exceptată, fiind constituită doar din fluturi de tip TFD_2.

Relațiile de calcul dintre eşantioanele lui v_{l+1} şi v_l (pe treapta $l+1$):

$$\begin{cases} v_{l+1}\left[2^{l+1}k+m\right]=v_l\left[2^{l+1}k+m\right]+w_{2^{l+1}}^{m}v_l\left[2^{l+1}k+2^l+m\right] \\ v_{l+1}\left[2^{l+1}k+2^l+m\right]=v_l\left[2^{l+1}k+m\right]+w_{2^{l+1}}^{2^l+m}v_l\left[2^{l+1}k+2^l+m\right] \end{cases}$$

$$\begin{cases} \forall l\in\overline{1,L-1} \\ \forall m\in\overline{0,2^l-1} \\ \forall k\in\overline{0,2^{L-l-1}-1} \end{cases}$$

Schema generică de calcul

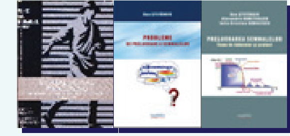


☝ Este permisă implementarea "in place".

☜ Două înmulțiri şi două adunări complexe!

Număr de operații

- **DFT :** $\mathcal{O}_0[N]=\left[4N^2\right]_{\bullet}+\left[2N(2N-1)\right]_{+}\sim 4N^2$
- **FFT-t :** $\mathcal{O}_2[N]=\left[4N\log_2 N\right]_{\bullet}+\left[4N\log_2 N\right]_{+}\sim 4N\log_2 N$



❼.❸ Algoritmul bazat pe segmentarea în timp (FFT-t)

A doua variantă de calcul, eficientă

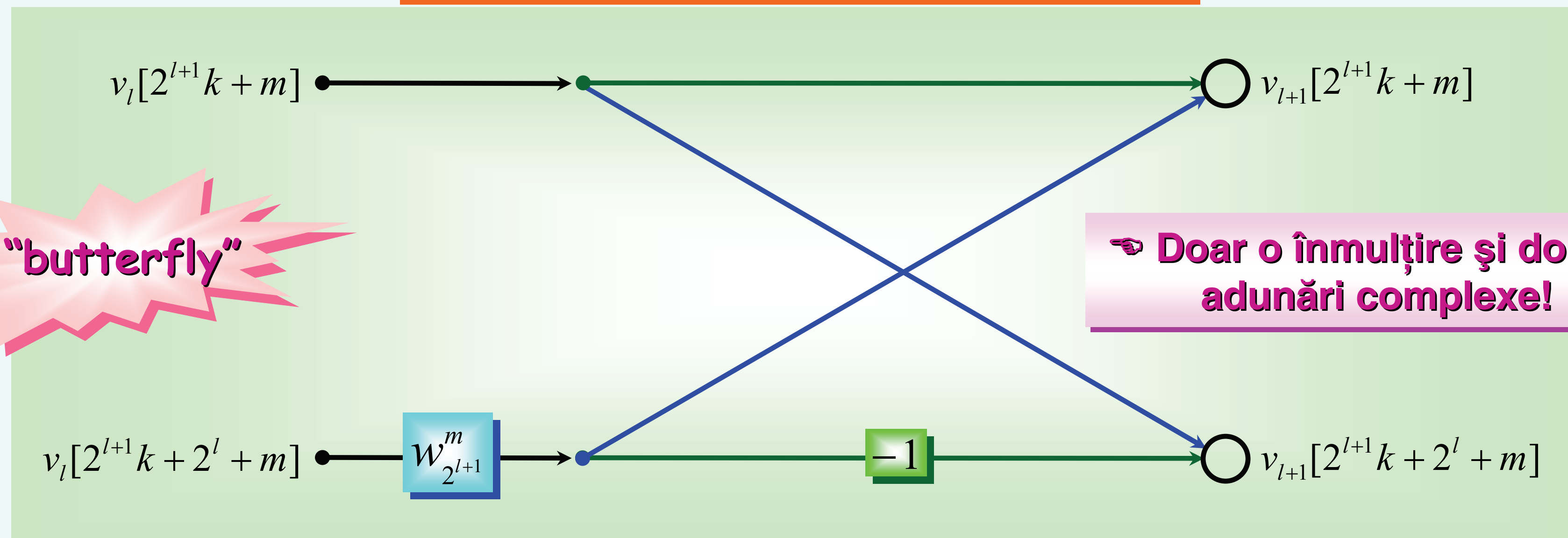
Relații de calcul între semnale intermediare succesive (pe treapta l+1)

$$\begin{cases} v_{l+1}\left[2^{l+1}k+m\right] = v_l\left[2^{l+1}k+m\right] + w_{2^{l+1}}^{m} v_l\left[2^{l+1}k+2^l+m\right] \\ v_{l+1}\left[2^{l+1}k+2^l+m\right] = v_l\left[2^{l+1}k+m\right] - w_{2^{l+1}}^{m} v_l\left[2^{l+1}k+2^l+m\right] \end{cases}$$

$$\begin{cases} \forall l \in \overline{1, L-1} \\ \forall m \in \overline{0, 2^l-1} \\ \forall k \in \overline{0, 2^{L-l-1}-1} \end{cases}$$

Antisimetrie: $w_{2M}^{M+m} = -w_{2M}^{m}$

Schema generică echivalentă de calcul



☜ Doar o înmulțire și două adunări complexe!

Număr de operații

DFT : $\mathcal{O}_0[N] = \left[4N^2\right]_{\bullet} + \left[2N(2N-1)\right]_{+} \sim 4N^2$

FFT-t : $\mathcal{O}_2[N] = \left[2N\log_2 N\right]_{\bullet} + \left[2N\log_2 N\right]_{+} \sim 2N\log_2 N$

Exemplu $N = 1024 = 2^{10}$: $\begin{cases} \mathcal{O}_0[N] \sim 2^{22} \\ \mathcal{O}_2[N] \sim 10\times 2^{11} \end{cases}$