520.492 Mixed-Signal VLSI Systems

Week 8

Vector Quantization

References

1. J.R. Deller, J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, New York: MacMillan, 1993
2. T. Kohonen, *Self-Organization and Associate Memory*, New York: Springer-Verlag, 3rd ed., 1989.
3. G. Cauwenberghs and V. Pedroni, "A Low-Power CMOS Analog Vector Quantizer," *IEEE Journal of Solid-State Circuits*, vol. 32 (8), pp. 1278-1283, 1997.
4. J. Lubkin and G. Cauwenberghs, "VLSI Implementation of Fuzzy Adaptive Resonance and Learning Vector Quantization," *Int. J. Analog Integrated Circuits and Signal Processing* (Kluwer ALOG), vol. 30 (2), pp. 149-157, 2002.

SCALAR QUANTIZATION

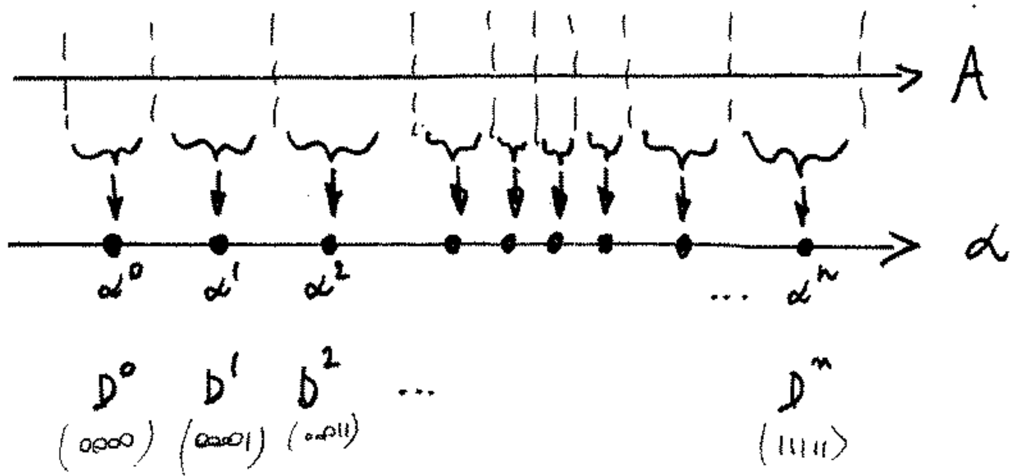
$A \longrightarrow \boxed{Q} \longrightarrow D$ $\qquad$ $D \longrightarrow \boxed{Q^{-1}} \longrightarrow \alpha$

coding / quantization $\qquad$ $D \in \{D^i\}$ $\qquad$ decoding / reconstruction $\qquad$ $\alpha \in \{\alpha^i\}$ → TEMPLATES

A: input field : (analog)

α: output field : (discrete analog) "TEMPLATES"

D: Coding : (digital

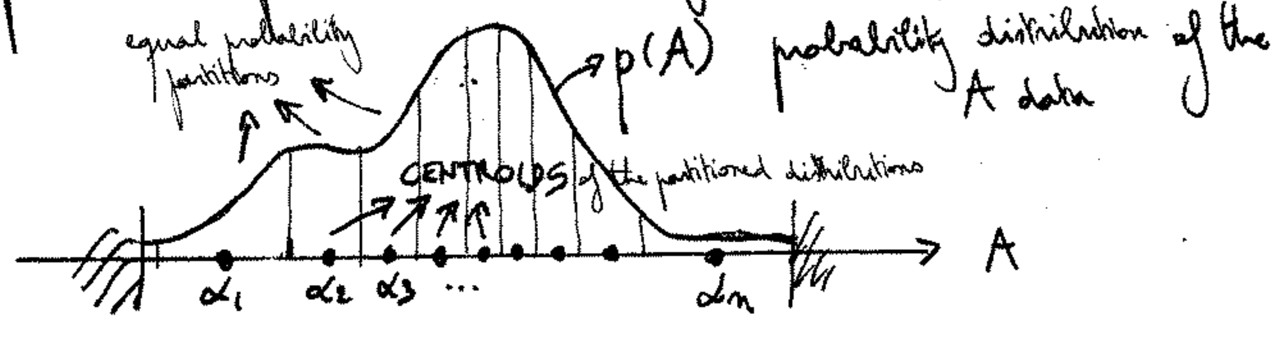


- basically: A/D conversion (though not necessarily uniform)
- more generally: NEAREST NEIGHBOR CLASSIFICATION
  TEMPLATE MATCHING

$$\boxed{D = \arg\min_{D^i} d(A, \alpha^i(D^i))}$$

d: distance metric

- optimal template set α^i: (for data compression)

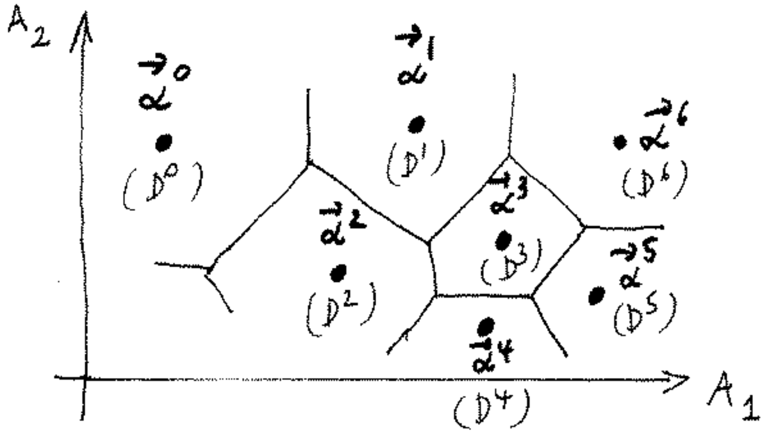


VECTOR QUANTIZATION

$$\vec{A} \longrightarrow \boxed{Q} \longrightarrow D \qquad\qquad D \longrightarrow \boxed{Q^{-1}} \longrightarrow \vec{\alpha}$$

coding quantization $\qquad D \in \{D^i\} \qquad$ decoding reconstruction $\qquad \vec{\alpha} \in \{\vec{\alpha}^{\,i}\}$

TEMPLATES

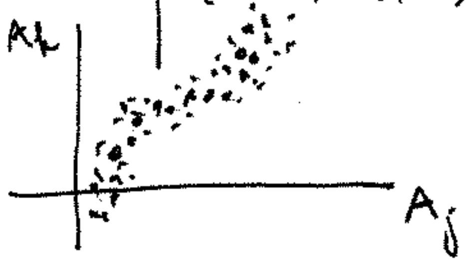


- multi-dimensional extension of scalar quantization
  (NOT : component-wise scalar quantization !)
- motivation: aims at reducing the data representation
  compact coding : DATA COMPRESSION (lossy)

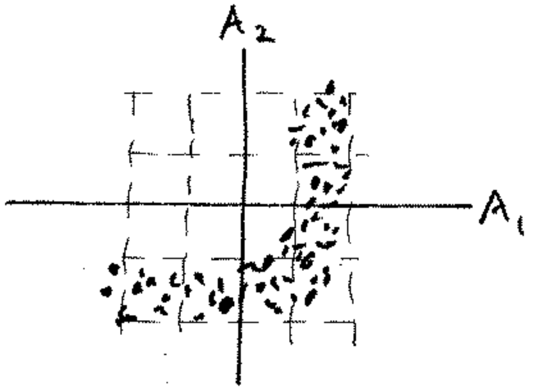
→ exploit [redundancy in the A_i data stream
  co-dependence in consecutive data points

"memory effects" (in temporal sequences ; SOUND)
"correlations" (in spatial fields ; IMAGES)

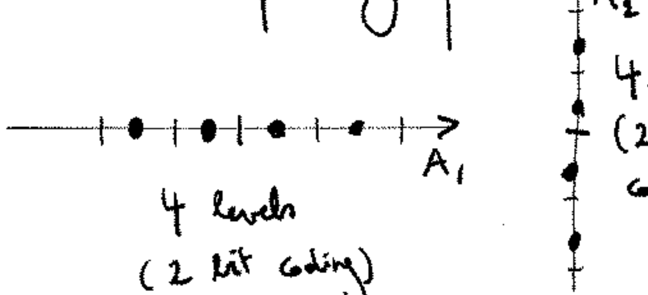
TYPICALLY: $p(A_1, A_2, A_3, \ldots) \cong p(f(A_1, A_2, A_3, \ldots))$



CLUSTERING in sub-dimensions of data space example:

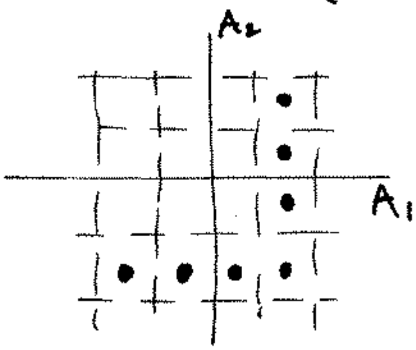


- A_1 and A_2 separately quantized (twice scalar quantization):



$\Rightarrow$ 16 possible outcomes (4 bit coding)

- A_1 and A_2 combined (vector quantization in 2-D):



7 levels (3 bit) $\Rightarrow$ 25% compression

(typically higher in more dimensions)

<u>Optimal partitioning</u> (positioning of the templates) for highest compression:

$\{\vec{\alpha}^{\,i}\}$ = centroids of equal probability partitions (as determined by $p(\vec{A})$ distribution)

(minimizes $\underset{\{A\}}{E}\left[d\left(\vec{A},\ \vec{\alpha}(\vec{A})\right)\right]$ for a given size of $\{\vec{\alpha}^{\,i}\}$ set)

CLUSTERING ALGORITHM:

$$\vec{A}^{(k)} \longrightarrow \boxed{Q} \longrightarrow \begin{matrix} D^{(k)} \\ \vec{\alpha}^{(k)} \end{matrix} \qquad \begin{matrix} D^{(k)} \in \{D^i\} \\ \vec{\alpha}^{(k)} \in \{\vec{\alpha}^i\} \end{matrix}$$

Training: Adjust the positions of the $\{\vec{\alpha}^i\}$ according to the incoming training data $\vec{A}^{(k)}$ $(k = 0, 1, \ldots)$:

For each $\vec{A}^{(k)}$:

- select the nearest $\vec{\alpha}^i$ (that is, $\vec{\alpha}^{(k)}$)
- update that $\vec{\alpha}^i$ (pertaining to) towards $A^{(k)}$:

$$\vec{\alpha}^i \Leftarrow \frac{m_i}{m_i+1} \cdot \vec{\alpha}^i + \frac{1}{m_i+1} \cdot \vec{A}^{(k)}$$

with m_i the number of times ^{i} has been previously selected (and updated)

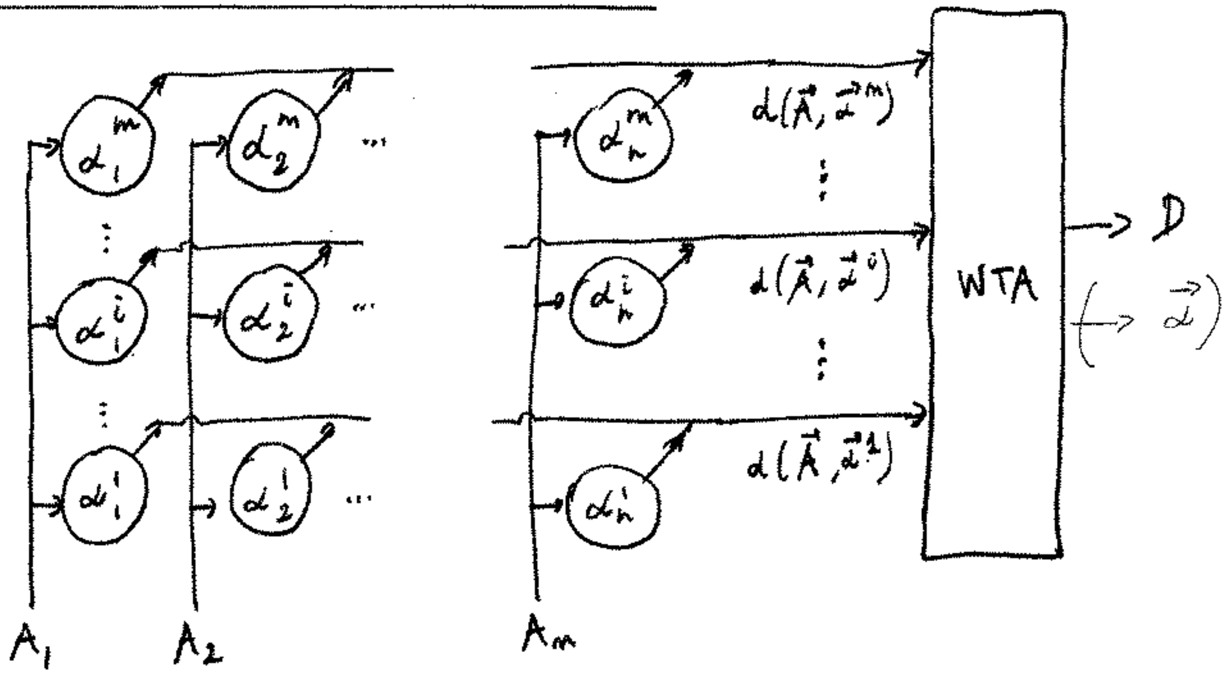
- continue with $\vec{A}^{(k+1)}$

Iteration over large training sets essentially clusters the data $\vec{A}^{(k)}$ in non-overlapping regions, each with a template $\vec{\alpha}^i$ located at the centroid of the partitioned distribution.

Extension: Kohonen maps: Topology preserving maps, by involving the spatially nearest neighbors along with the winner into the update process.

(Update $\vec{\alpha}^i \Leftarrow \frac{m_i}{m_i+1} \cdot \vec{\alpha}^i + \frac{1}{m_i+1} \vec{A}^{(k)}$ for all $\vec{\alpha}^i$ "near" the winner $\vec{\alpha}^{(k)}$)

PARALLEL IMPLEMENTATION



- distance calculation:

$$d(\vec{A}, \vec{\alpha}^i) = \sum_j |A_j - \alpha_j^i|^2$$

- winner-take-all selection:

$$\underset{(\vec{\alpha})}{D} = \underset{\substack{\{D^i\} \\ (\{\vec{\alpha}^i\})}}{\arg\min} \; d(\vec{A}, \vec{\alpha}^i)$$

- training:

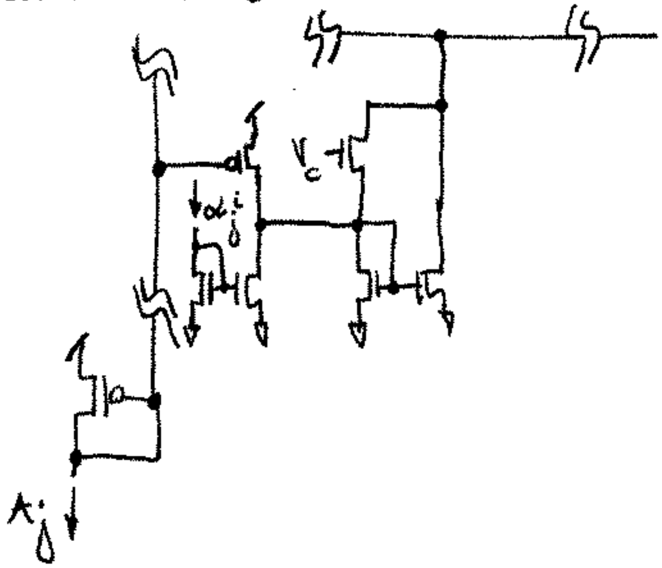
$$\alpha_j^i \longleftarrow (1-\lambda)\,\alpha_j^i + \lambda A_j \quad \text{only for } i = \text{winner}$$

(or close to the winner)

$(\lambda = \text{small})$

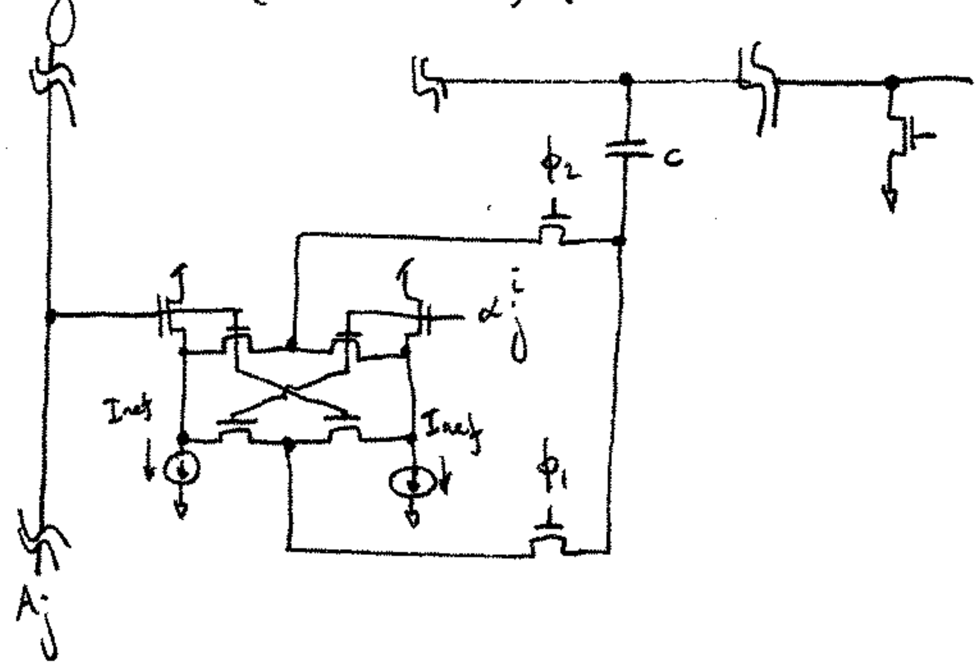
distance calculation:

• current-mode:



$\leftarrow \sum_{j=1}^{n} |A_j - \alpha_j^i|$ $(b=1)$

• voltage-mode (switched-C):

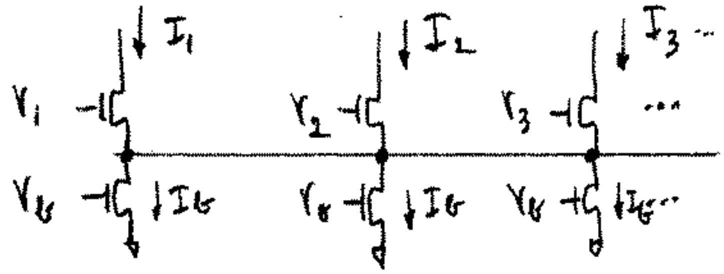


$\frac{1}{n}\sum_{j=1}^{n} |A_j - \alpha_j^i|$

$(b=1)$

Winner-take-all:

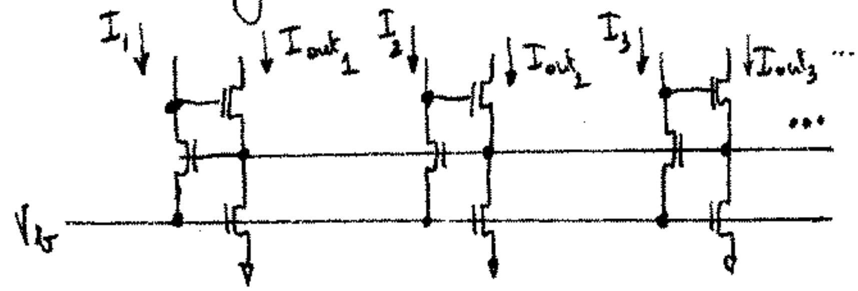
- simplest structure (voltage in, current out):



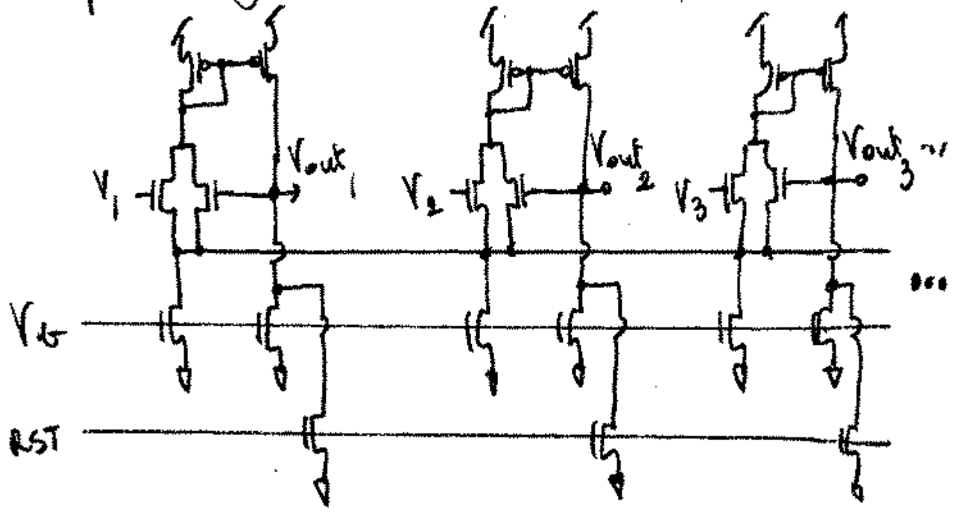
$$\Rightarrow \quad I_i = \frac{e^{\frac{\kappa V_i}{V_T}}}{\frac{1}{n}\sum_{j=1}^{n} e^{\frac{\kappa V_j}{V_T}}} \cdot I_b$$

$\approx$ ↗ $n I_b$; winner
↘ 0; the others

- increased gain, current mode:

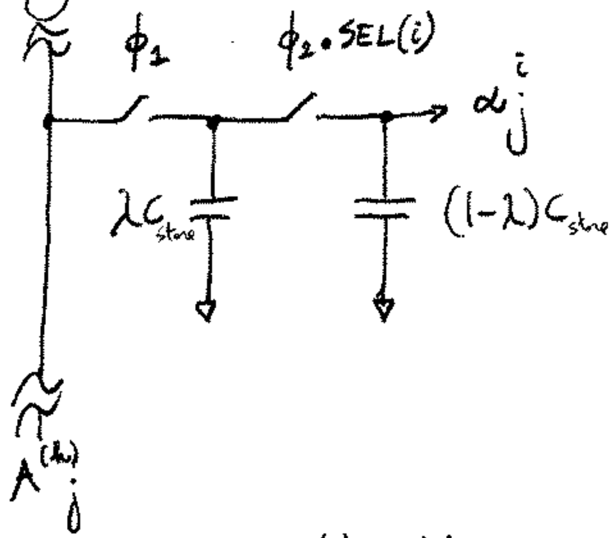


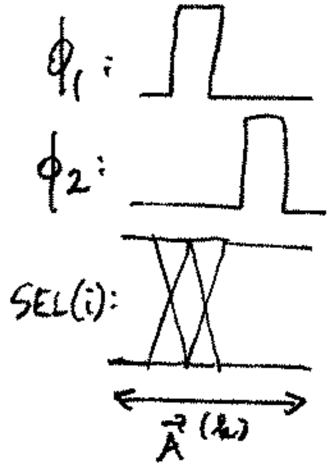
- positive feedback, voltage mode (triggered):



Training: (adaptation)

- voltage-mode distance calculation circuitry:

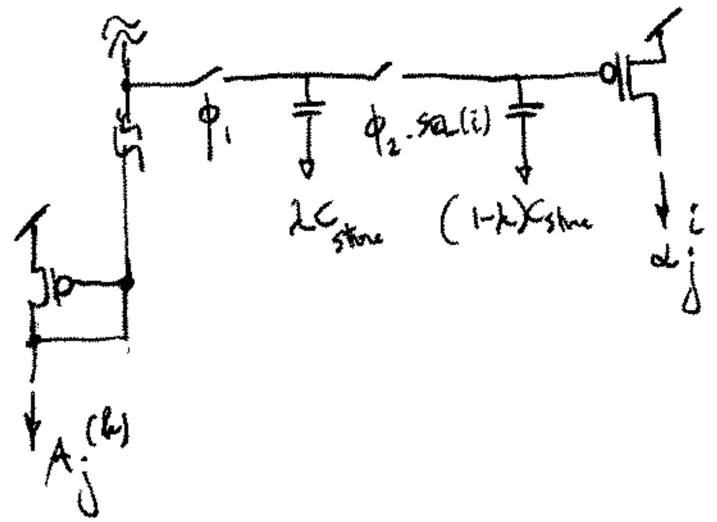




SEL(i): determined by the winner of the $\vec{A}^{(k)}$ matching

- k-means clustering : i = winner $\Rightarrow$ SEL(i) = TRUE
- Kohonen : i = winner $\Rightarrow$ $SEL(\underbrace{i \pm 1}_{\text{NEIGHBORHOOD}})$ = TRUE

- current-mode: same, but with gate voltages of current sources:



(ϕ_1 and $\phi_2 \cdot SEL(i)$ could be voltage range limited)