Welcome to

0510.7720.01 Winter semester 2021

Mixed Signal Electronic Circuits

Instructor: Dr. M. Moyal

**Lecture 11 Cont. after Lect 10 and back to 9(sar).**

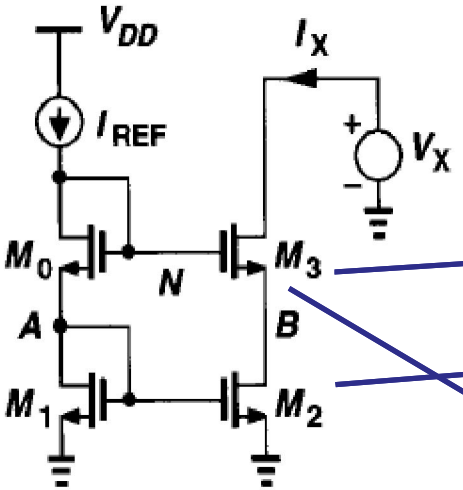
**18:10-- June 02 2021**

***OVERSAMPLING DIGITAL TO ANALOG CONVERTERS***

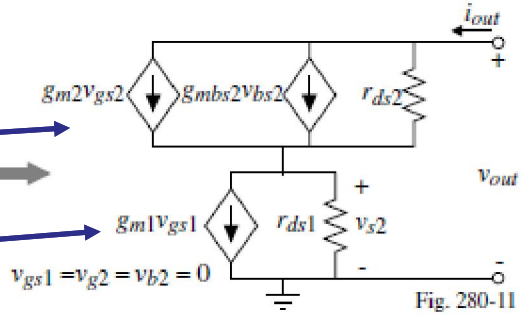
1. **Architectures Types – consideration**
2. **DWA , DEM – calibration architectures.**
3. **Examples**

**Some over view of lect. 10 and Cascade current source..**

Remember CMOS Tr. Is nothing but a current source (is sat). – most of the time..
So understanding its basic is key.. Rout=Vx/Ix ( small signal)




Step 3 – ignore my M..



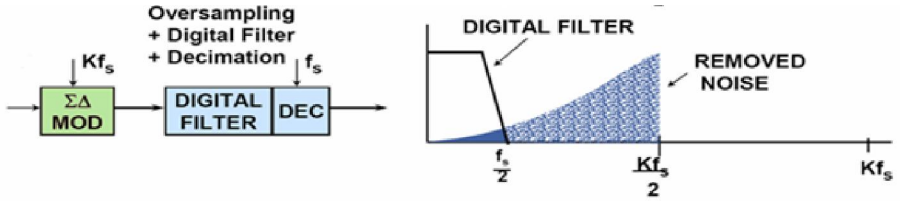

Step 1 – inspect the sizes (of who ? - if the same iref=ix)

Step 2 – inspect the voltages limits.. Approx..(understanding)

Step 3 – Build small signal or predict the Rout..

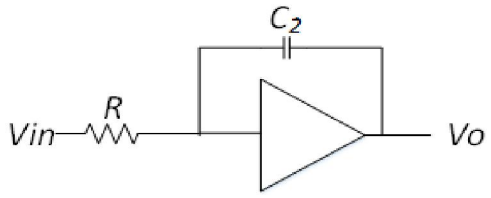
$$r_{out} = \frac{v_{out}}{i_{out}} = r_{ds1} + r_{ds2} + g_{m2}r_{ds1}r_{ds2} + g_{mbs2}r_{ds1}r_{ds2} \approx g_{m2}r_{ds1}r_{ds2}$$

CMOS Analog Integrated Circuit Design
Course - Allen

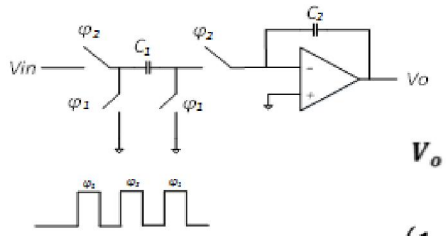



$$Z^{-1} = e^{-j\omega t}$$

***We will need digital filter..***




$$V_o = -\frac{1}{RC_2}\int_{-\infty}^{t} V_{in}(t)dt$$




***Gain***

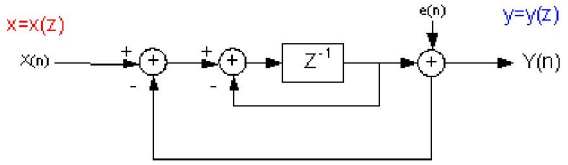
$$V_o = V_o(n-1) + \left(-\frac{C_1}{C_2}\right)V_{in}(n)$$

$$(1 - Z^{-1})V_o = -\left(\frac{C_1}{C_2}\right)V_{in}(n)$$

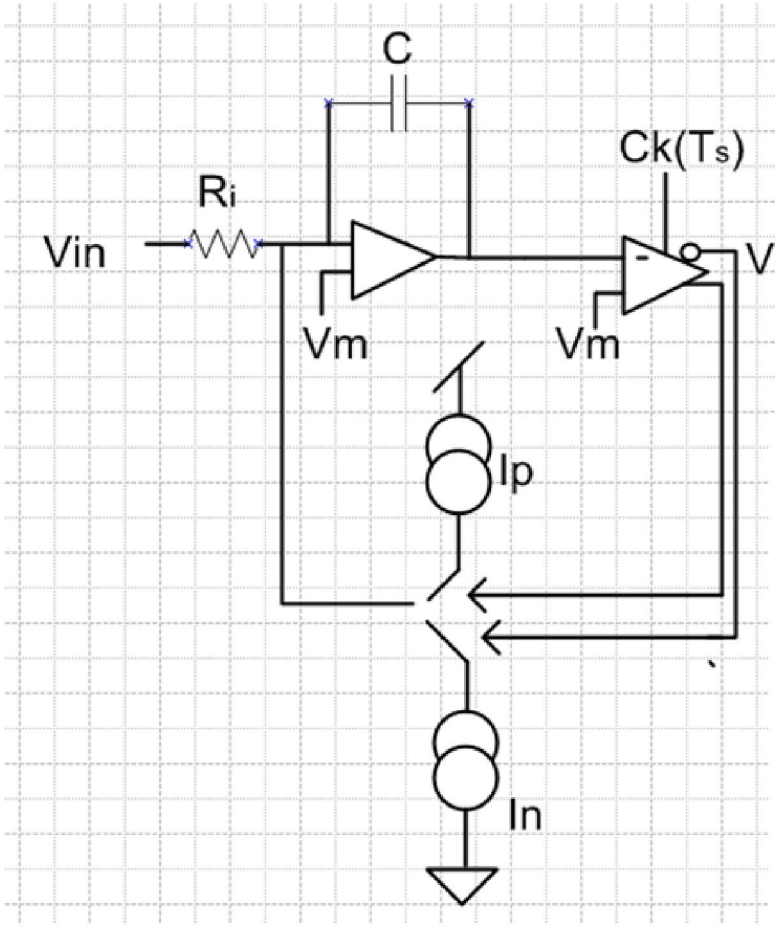
$$H(z) = \frac{-C_1/C_2}{1 - Z^{-1}}$$

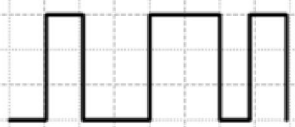
$$Y(z) = z^{-1}X(z) + (1 - z^{-1})E(z)$$

$$\vdots \int_0^{fb} \frac{\sigma_e^2}{fs}\left|1 - e^{-j\omega t}\right|^2 df$$

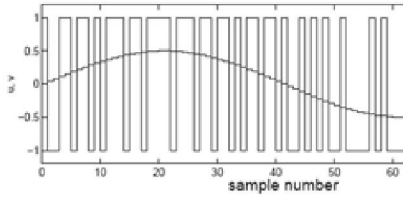


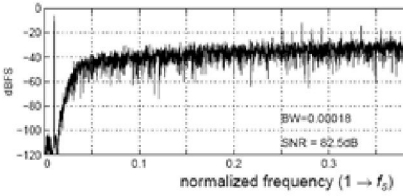

1st order SD model



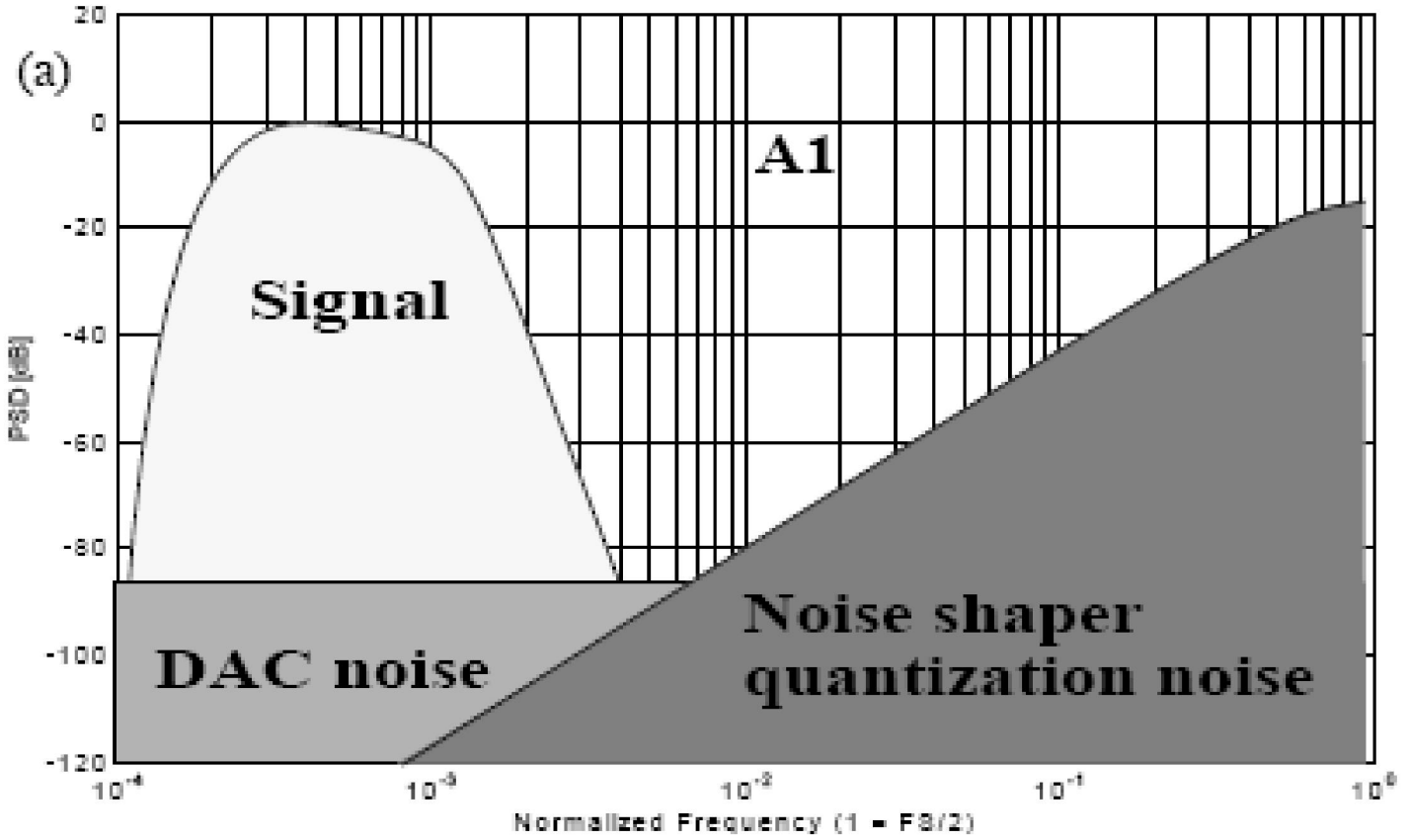



**Example Simulation**



# Concept of over sampling DACs

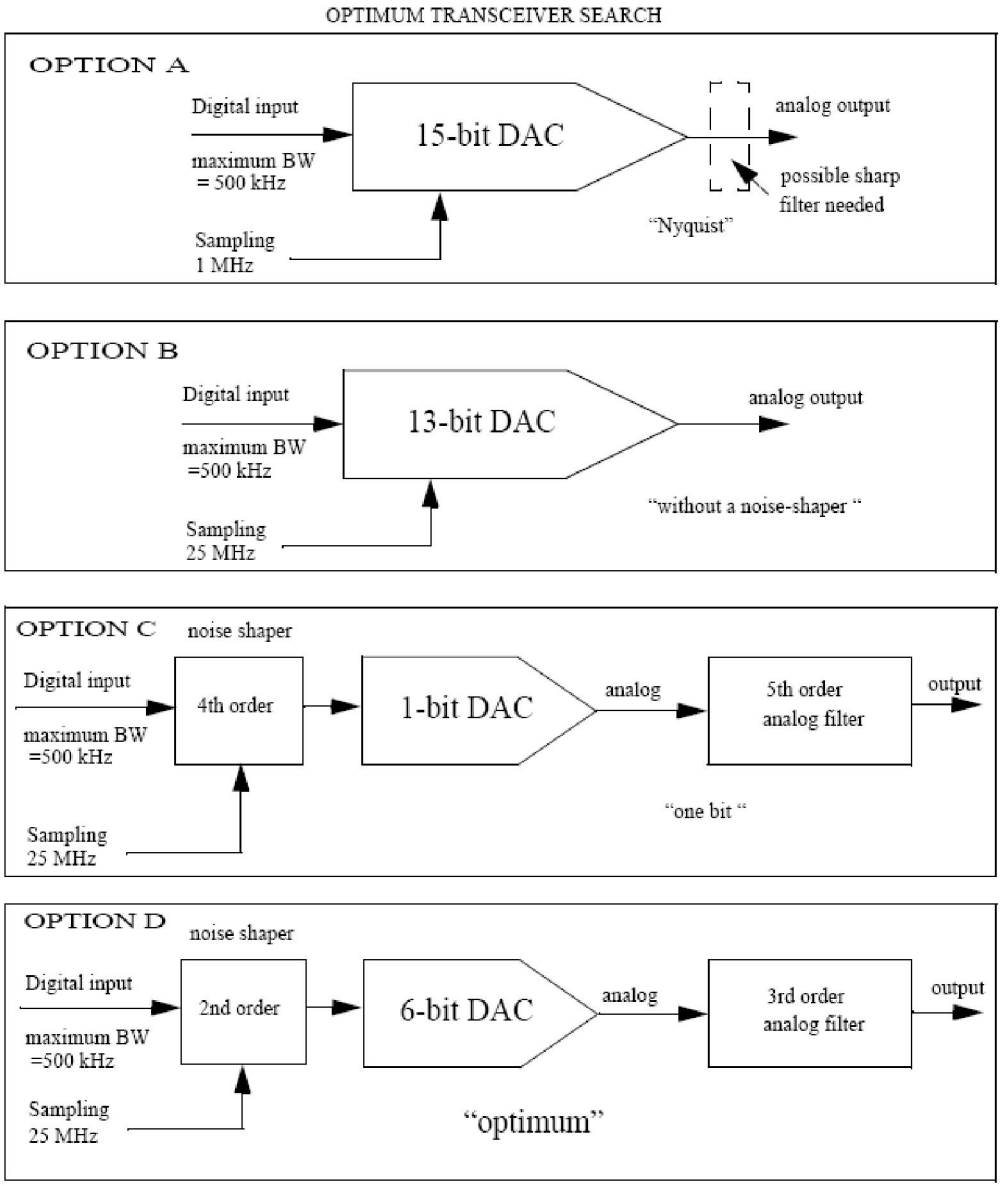



***DAC noise is Thermal Noises from DAC reference and circuit elements – It limits the Dynamic range***

ADDED Lecture.. For 10)

# Over Sampling DACs
# Error Feedback – Noise Shapers
# DAC Consideration Example

# Noise Shaping: 1st Order



From Fig. 23a, the output Y(z) is derived. The term T(z) at node (1) is given by:

$$T_{(z)} = e_{(z)}Z^{-1} + X_{(z)} \qquad \textbf{(5.14)}$$

Where: e(z) are the truncated bits and X(z) is the input signal. T(z) can also be written as:

$$T_{(z)} = Y_{(z)} + e_{(z)} \qquad \textbf{(5.15)}$$

By subtracting Eq. (5.14) from (5.15) and combining the e(z) terms, T(z) is eliminated

$$Y_{(z)} = X_{(z)} - (1 - z^{-1})e_{(z)} \qquad \textbf{(5.16)}$$

The truncating error is shaped by the function $1\text{-}z^{-1}$. This function is referred to as the Noise Transfer Function (NTF(z)). From Eq. (5.16), it is the coefficient of –e(z). hence

$$NTF_{(z)} = 1 - z^{-1} \qquad \textbf{(5.17)}$$

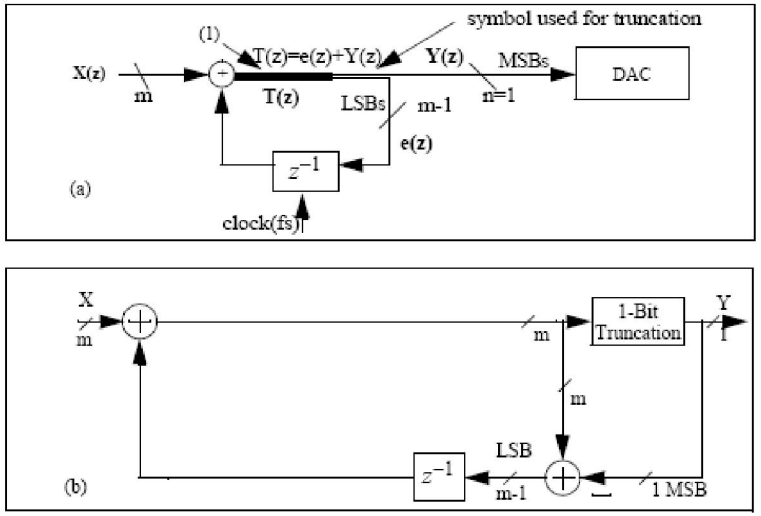


Figure 23: First order Single bit NS: (a) principle (b) practical design

## An Example of 1st Order NS

To demonstrate the effectiveness of this method, let us use an example with 3-bit input and 1 bit truncation (n=2). The input is assumed to be the binary number 101 (integer 5, m=3). Using Fig. 24, the output sequence, $Y$, is computed for each clock period as follows:

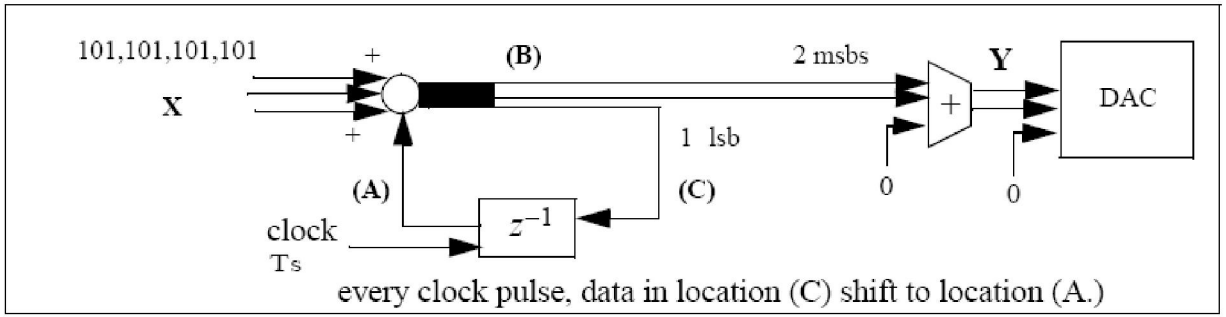



Figure 24: "DC" 101 input to a first order, 2-bit NS

| | X (input) | (A) | (B) | (C) (truncated) | Y (output) |
|---|---|---|---|---|---|
| T1 | 101 | 000 | 101 | 001 | 100 |
| T2 | 101 | 001 | 110 | 000 | 110 |
| T3 | 101 | 000 | 101 | 001 | 100 |
| T4 | 101 | 001 | 110 | 001 | 110 |

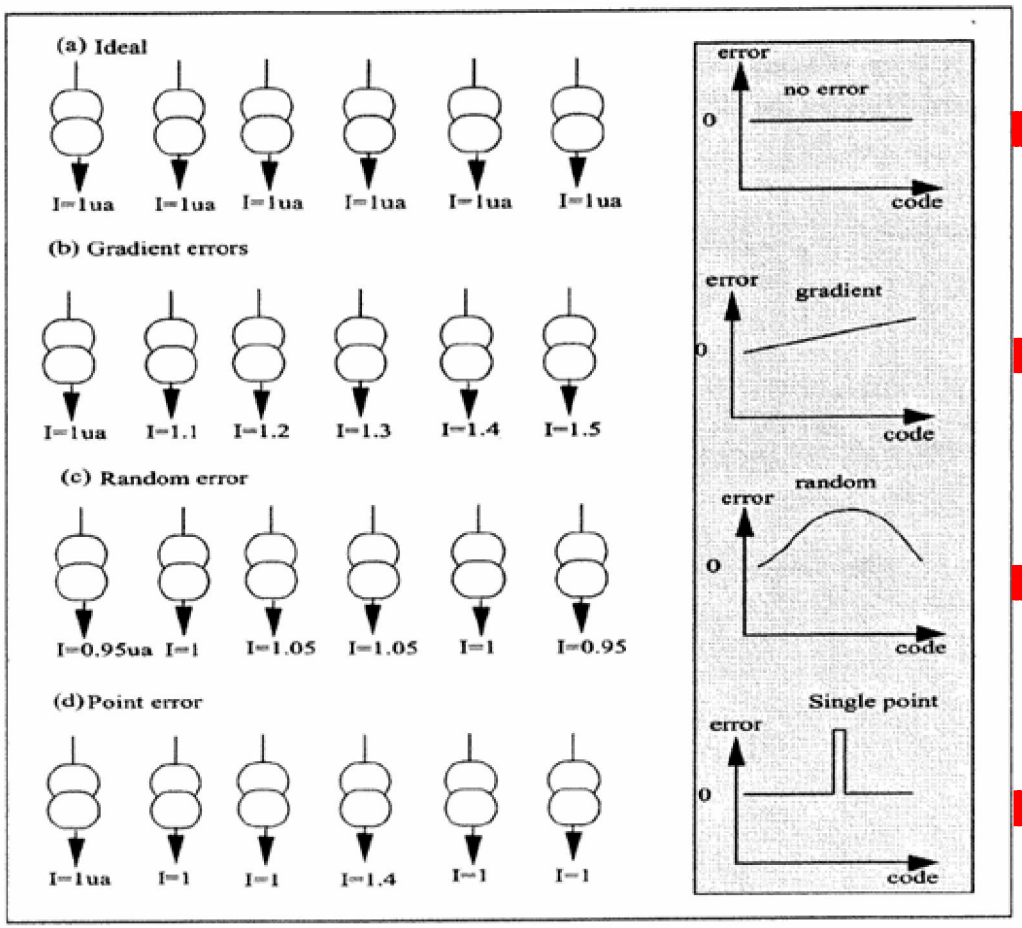
The final output is the average of all outputs, which are 100 and 110. For the sequence of 4,6,4,6, the average is the number "5."

# DAC Calibration Techniques

## DEM Method

DEM (Plassche 76, Galton 95) randomize the DAC bit by rotating and filtering the elements/bit much faster then the input signal.

## Mismatches of Current Source




No Error

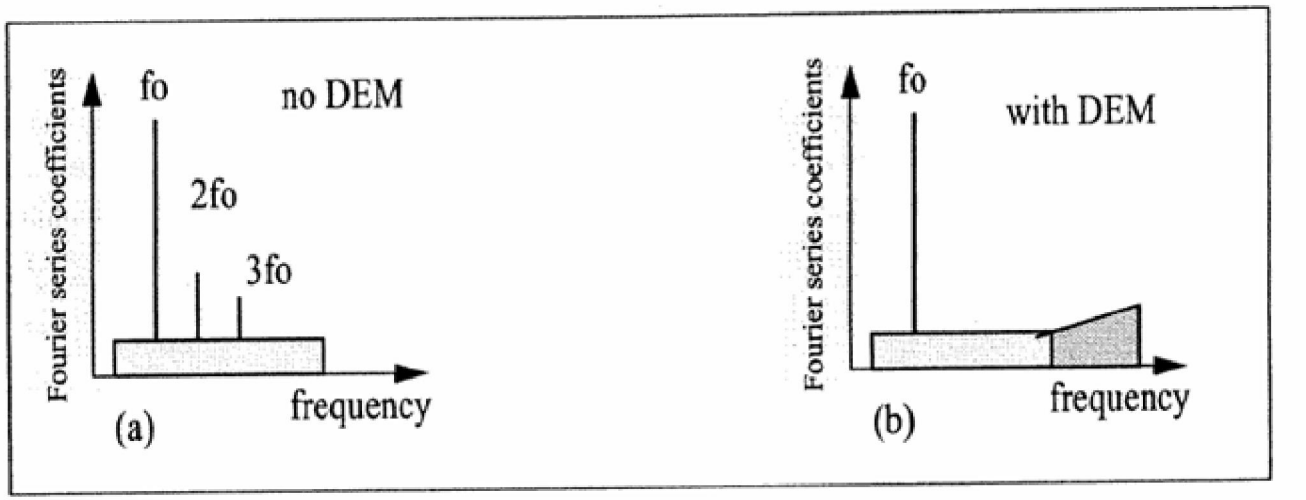
Gradient

Random

Single point

## DEM – Continue

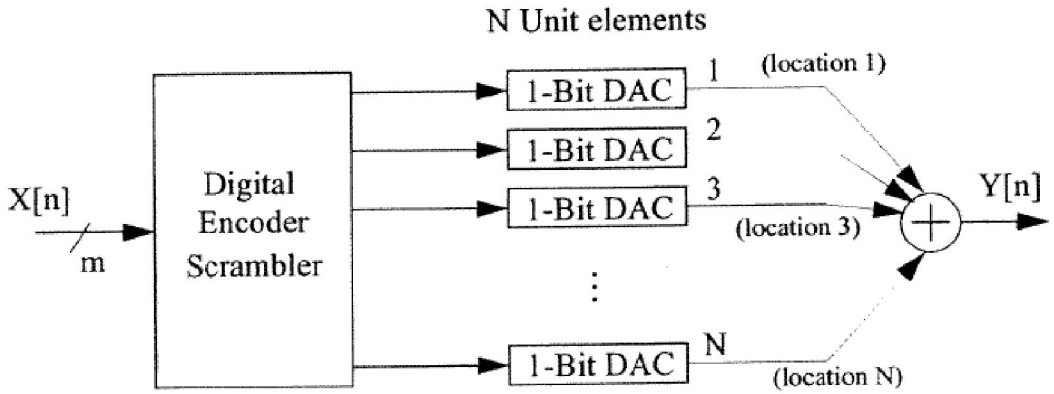
- **Power spectral density input is a sign wave**




(a) No DEM (b) with DEM

- Noise results from the DEM is less important as long as it lies outside the BW of the data.
- The smoothing filter and the line after the DAC will reduce these noise components

# DAC Model with Averaging




### 7.5. DEM Algorithms

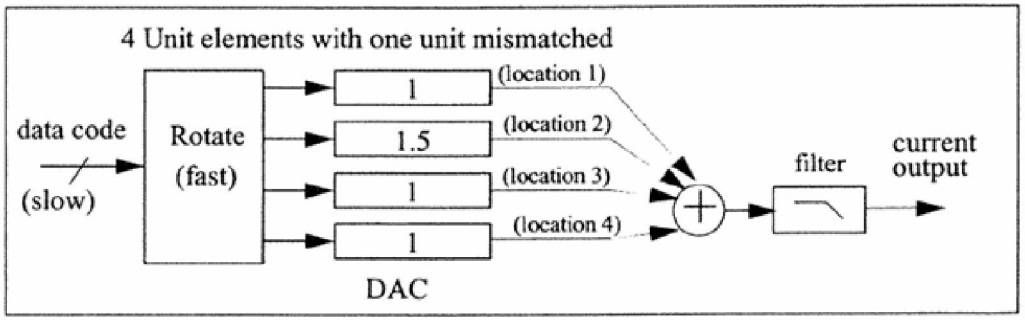
Several DEM techniques have been proposed, e.g., the so-called *Barrel-shift*, *data weighted averaging* (DWA) algorithms, and *random averaging*. For a brief explanation, imagine the unit elements are arranged on a circle. The algorithms mentioned above use a pointer on the chain of unit elements to identify the starting point for the sequence of elements that have to be selected. The algorithms differ in the way the pointer addresses during the next conversion interval.

The *barrel-shift* simply increases the pointer address by one until the end of the unit element chain is reached and the pointer is reset to the first. In contrast, the DWA algorithm calculates the new pointer address depending on the value of the current data to be converted by increasing the address by exactly the number of selected units. The address is calculated by a modulo-$N$ operation that takes care of the case when the pointer address exceeds the maximum $N$.

Furthermore, assuming ideal unit elements, the $k$ unit-elements for a given digital input signal are chosen and the output will always be the same. This shows that the scrambler within the digital encoder does not have an effect on the input signal component but modulates the DAC noise component of Y[n].

# DEM Example

Error in second unit of 50%




for digital code 00 , the output remains zero, for digital input code 01 , the output is:

$$Idac = (1+1.5+1+1)/4 = 4.5/4,$$

for digital input code 10 , two adjacent currents are selected and rotated along the array, and the output is given as

$$Idac = ((1+1.5)+(1.5+1)+(1+1)+(1+1))/4 = 9/4,$$

Finally, for digital input code 11 , three current units are selected, and rotated, and the output is given as

$$Idac = ((1+1.5+1)+(1.5+1+1+)+(1+1+1)+(1+1+1.5))/4 = 13.5/4$$

As illustrated, the values 4.5, 9, and 13.5 are linear multiples of the input code. Therefore, non-linearities are cancelled via the rotations and the filtering action. The *Barrel-shift* algorithm does not depend on the type of mismatch as long as there is enough time for the *Barrel-shift* algorithm to go around the DAC elements completely. Also, because the DAC errors are repeated every $N$ samples, the noise power using the *barrel-shift* algorithm is concentrated around a frequency given as

$$f_{BS} = \frac{f_S}{N} \qquad (7.1)$$

where $f_S$ is the sampling frequency. Consequently, the number of bits of a DAC using the *Barrel-shift* algorithm must be as small as possible given the maximum sampling frequency.

# 1st Order DWA Algorithm

In this algorithm, the rotation jump is correlated with the input data. If the input is 5, then the next jump is by 5, and five current sources are selected. In addition, if the input is 7, then the next jump location will also be 7.

As an example, DWA is applied to a 2-bit DAC having $N = 4$. If we let the current elements I+e1, I+e2, I+e3, and I+e4 be the DAC current units at locations 1,2,3, and 4, where e1-e4 are the DAC current mismatch errors, and we define e1+e2+e3+e4=$es$, then for the input 01, the output remains the same as the *barrel-shift* case, and is given as

$$Idac = \frac{4I + e1 + e2 + e3 + e4}{4} = 1\left(\frac{4I + es}{4}\right), \quad \text{cancellation cycle=4.}$$

For input data: 10 $Idac = \frac{(2I + e1 + e2) + (2I + e3 + e4)}{2} = 2\left(\frac{4I + es}{4}\right)$, cancellation cycle=2, and for input data 11 (locations: 123, 412, 341, and 234 are grouped):

$$\frac{(3I + e1 + e2 + e3) + (3I + e4 + e1 + e2) + (3I + e3 + e4 + e1) + (3I + e2 + e3 + e4)}{4} = 3\left(\frac{4I + es}{4}\right)$$

cancellation cycle=4 which results in the worst case cancellation cycle.

The advantage of the DWA algorithm is that averaging occurs more frequently. For every input which is an integer multiple of the total number of current sources, the averaging cycle is reduced. As example, for input code of 10, cancellation cycle is 4/2=2.

Mathematical analysis shows that the error signal from a DWA DAC is first order shaped, which is filtered by $1 - z^{-1}$ [Galton, Carbone95]. Other algorithms, like a second order DWA perform a second order high-pass shaping.

## Random Algorithm

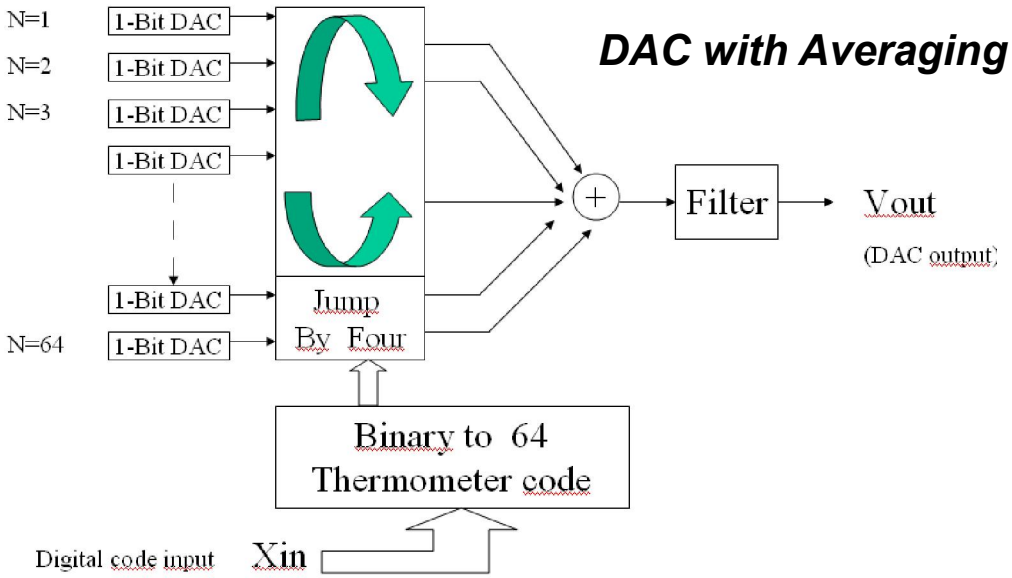
The signal index for this algorithm is selected randomly. All of the harmonic distortion components have been translated into white noise that partially falls into the baseband, therefore, reducing *SNR*. Simulation results show that this case does not produce optimum cancellation. It is impossible to produce a pure random signal. Eventually, it will repeat itself with a frequency that falls within the signal bandwidth.

### d) New Hopping" algorithm - final implementation

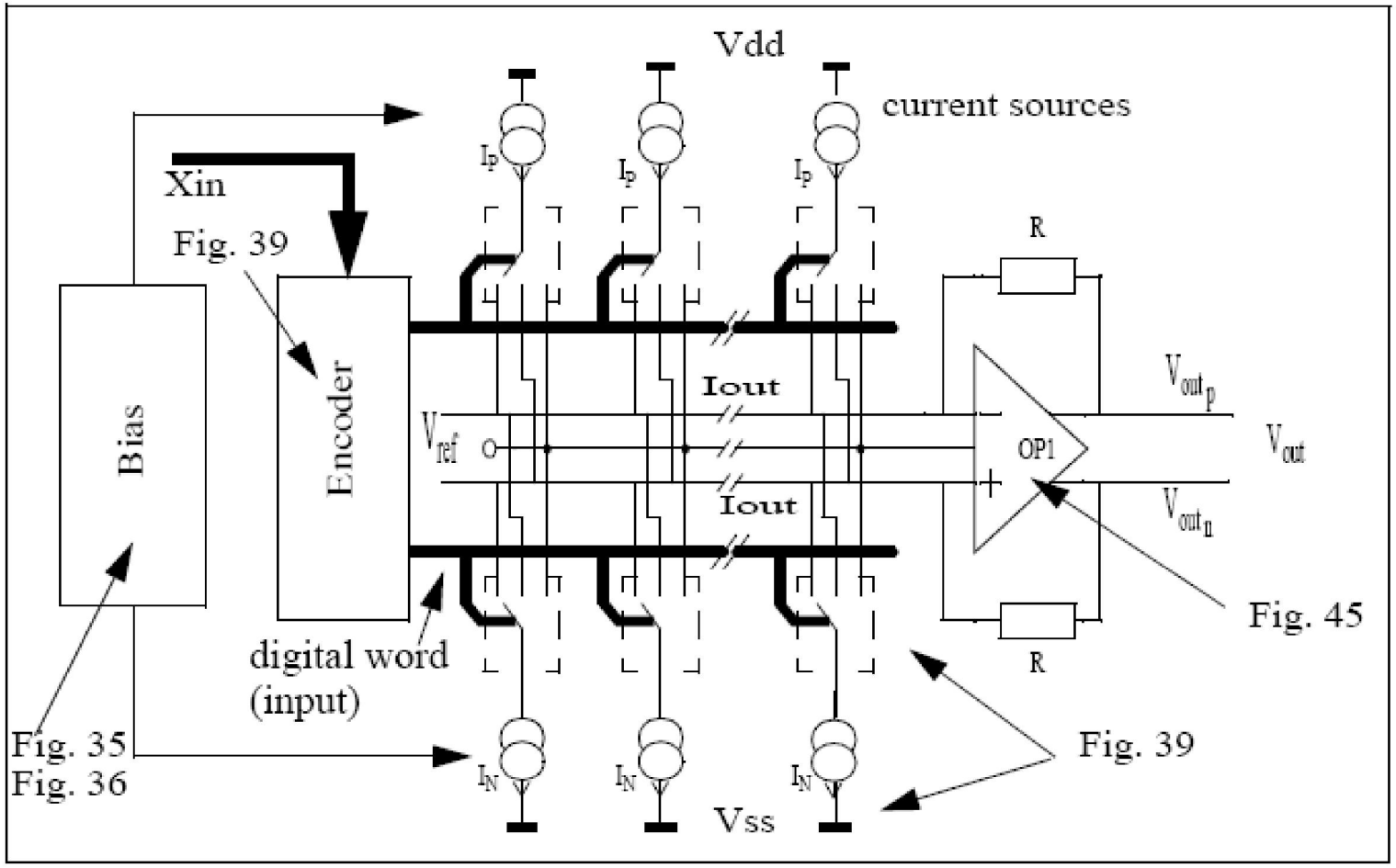
The hopping" algorithm reported at *ISSCC99* [Moyal et al. 99] is a modified *Barrel-shift* algorithm developed to avoid the long cycle time. A jump of four was chosen, because the frequency by which the errors are repeated can be increased by using larger shift steps (e.g., four instead of one, during the calculation of the next pointer address).

## Modifies Averaging

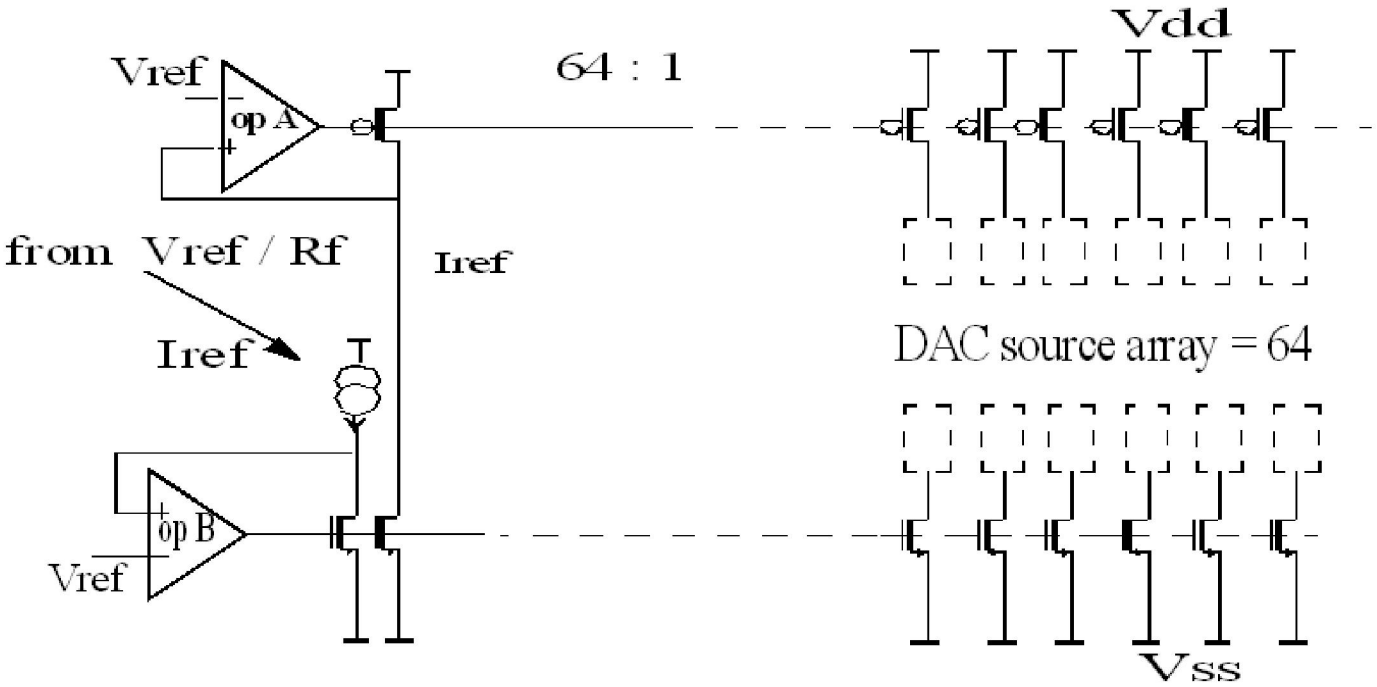
### USE A CONSTANT JUMP OF 1 ( 64 units )



# DAC – Differential Architecture – An Example



## DAC Bias



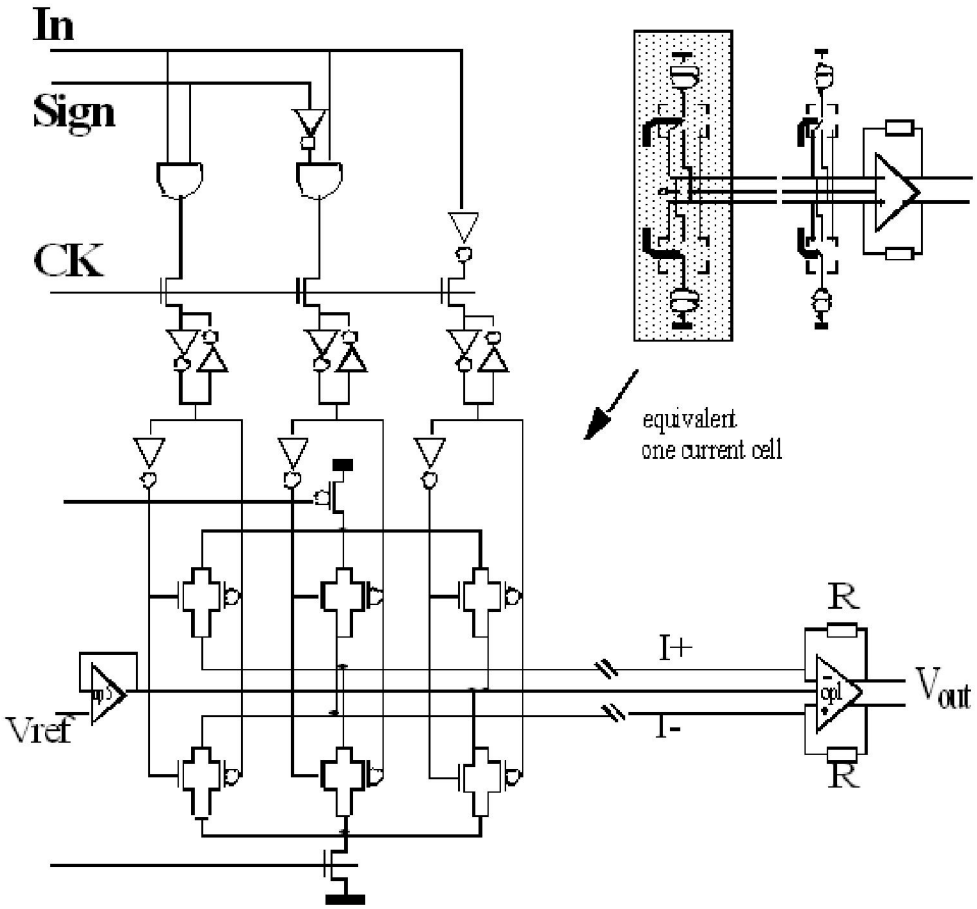

- Op B and op A outputs drive the DAC array

Source: M Moyal

## LSB Implementation

- 64 units
- **Sign** Input is for Current direction
- **CK** is to Latch the Data
- An Example:

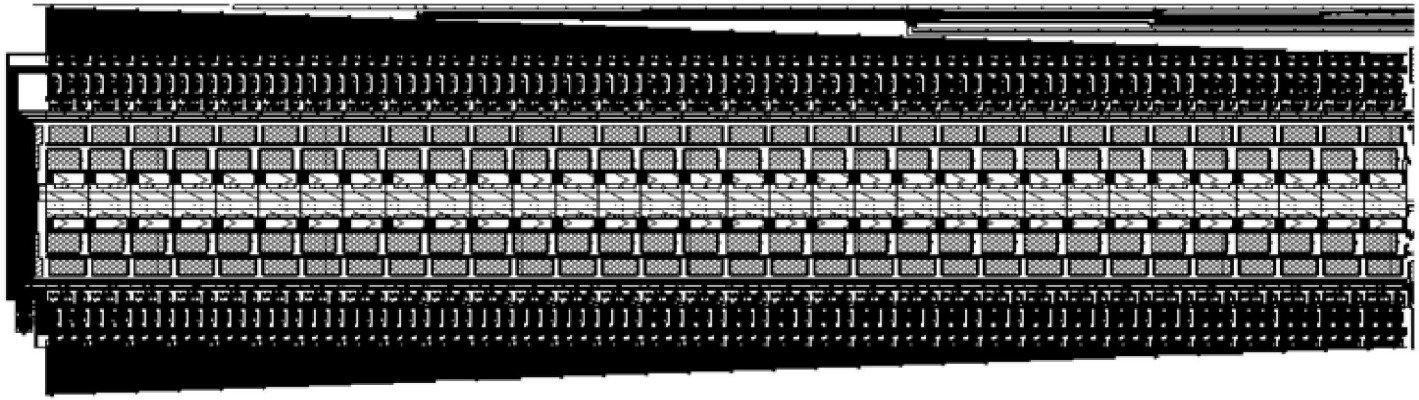
  If **In** = 0 → Vout = 0




Source: M Moyal

## DAC Layout

- Hand layout to allow "shielding" of analog from digital
- Iout lines are in the middle
- Digital on the outside
- Area: Core < 0.6 $mm^2$ ( total <1)

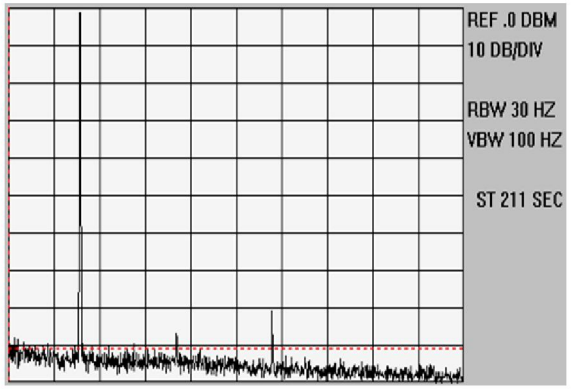


# Measured Results

## Transmitter Harmonic

WITHOUT DYNAMIC AVERAGING

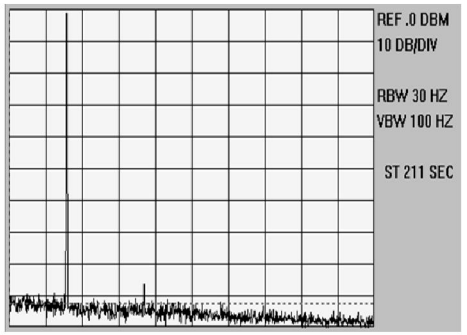
2rd Harmonic at - 87dB
3nd Harmonic at - 78dB
5th Harmonic at - 87dB




## Transmitter Harmonic Cancellation

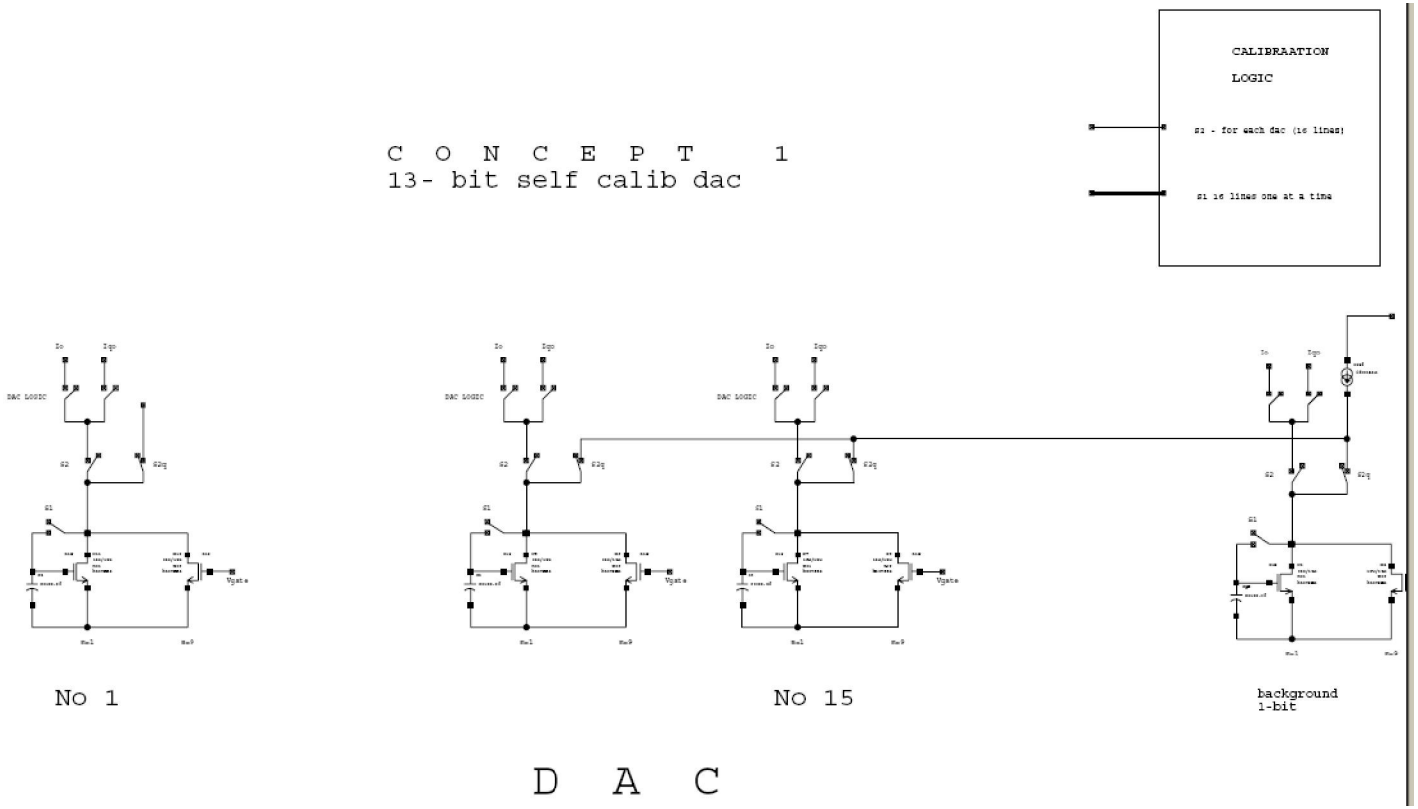
WITH DYNAMIC AVERAGING

2rd Harmonic at - 87dB
3nd Harmonic at - 93dB
5th Harmonic at - 100dB



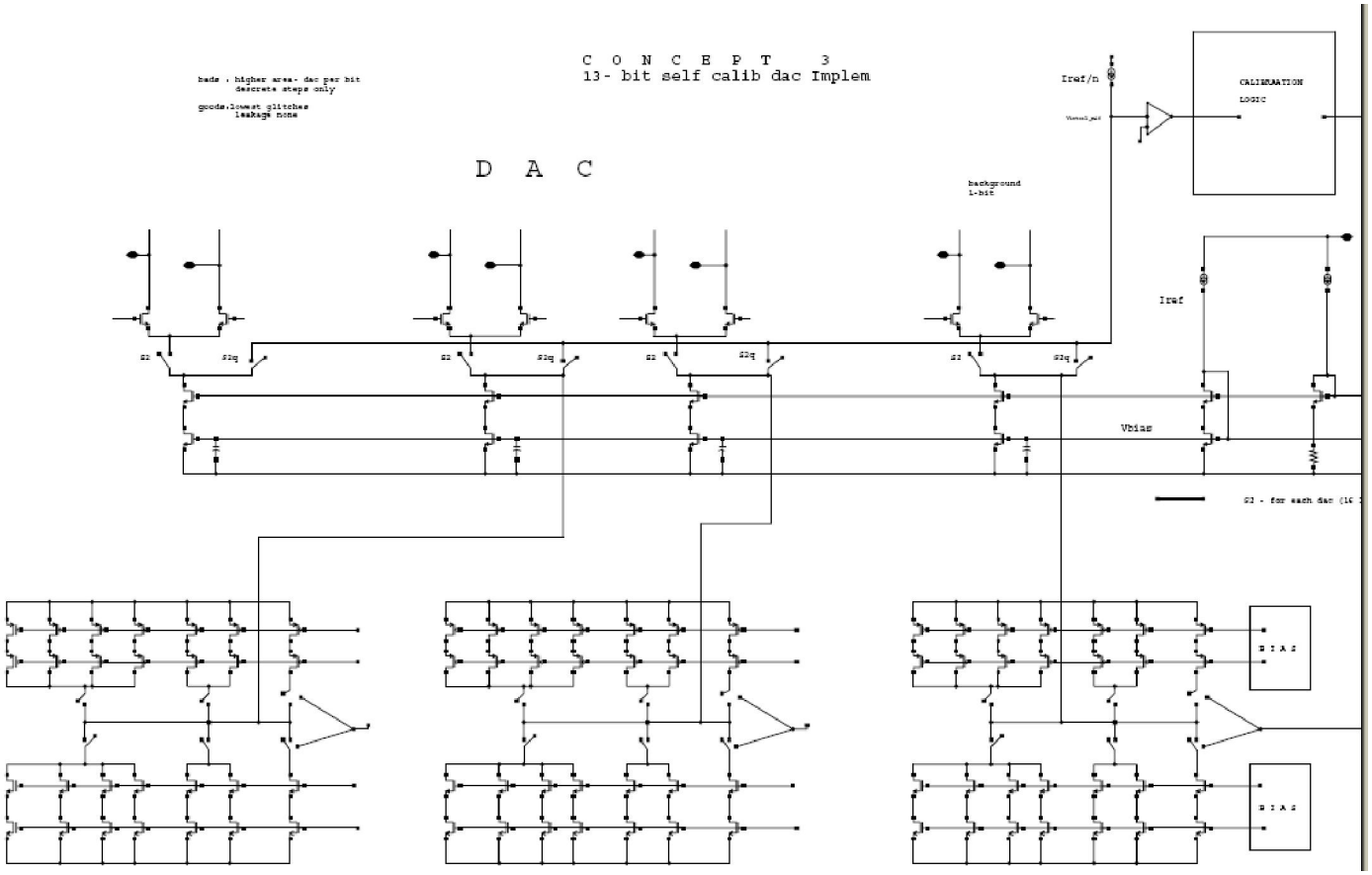
# DAC CALIBRAION TECHNIQUES

# Method I




- Calibration can be done during normal operation
- Keep the rate slow out of band but not to slow to have the capacitor discharge
- Switches error can be a problem

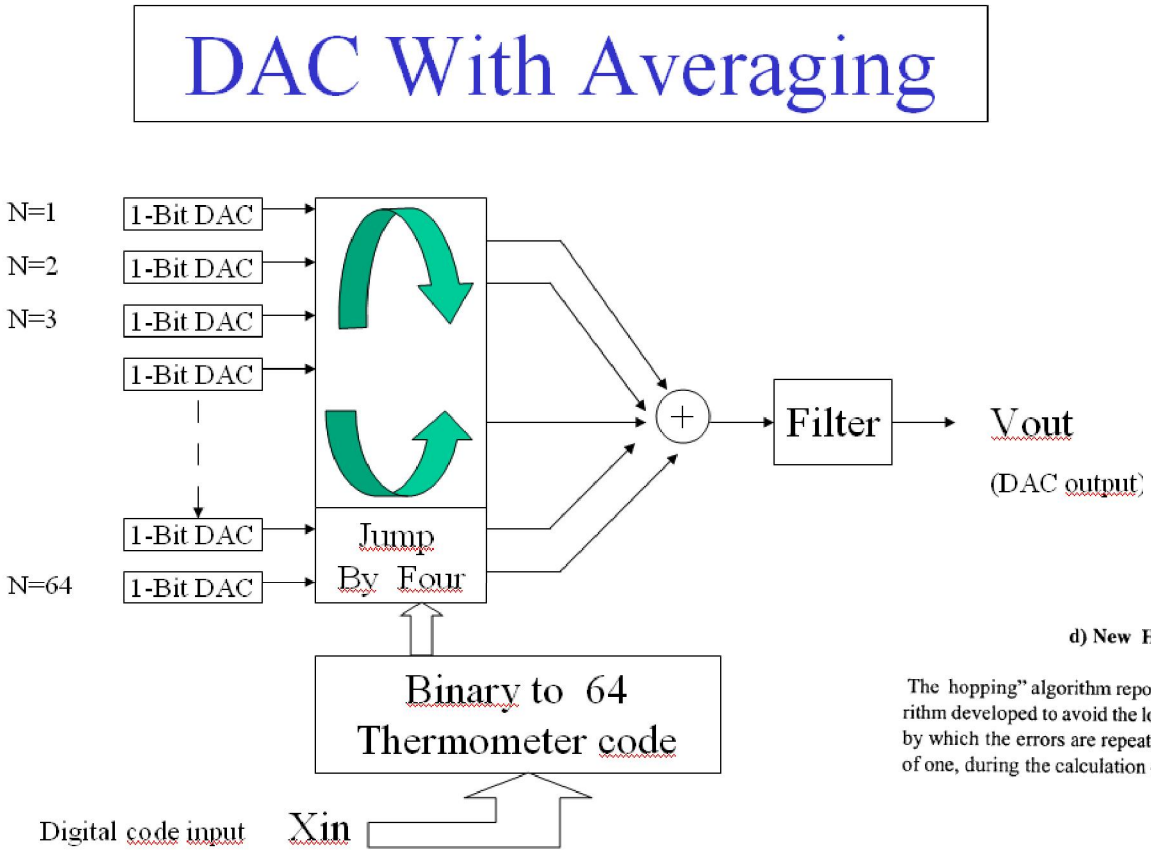
## Method II




- Attractive way
- Routing and layout is important
- Discrete steps

# MODIFIES AVERAGING

## USE A CONSTANT JUMP OF 4 ( 64 units )




**d) New Hopping" algorithm - final implementation**

The hopping" algorithm reported at *ISSCC99* [Moyal et al. 99] is a modified *Barrel-shift* algorithm developed to avoid the long cycle time. A jump of four was chosen, because the frequency by which the errors are repeated can be increased by using larger shift steps (e.g., four instead of one, during the calculation of the next pointer address).

# End of Lecture 06