

# Lecture 21

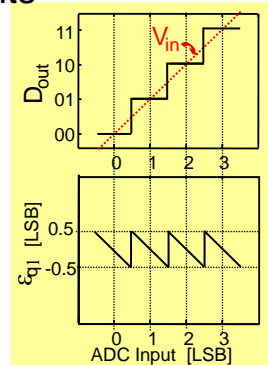
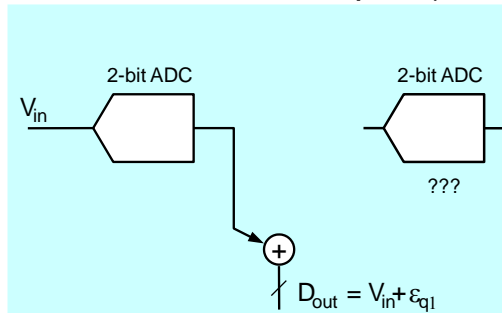
## Analog-to-Digital Converters (continued)

- Residue Type ADCs
  - Two-Step flash
  - Pipelined ADCs
    - Concept and basics of the architecture
    - Effect of building block non-idealities on overall ADC performance
      - Sub-ADC
      - Sub-DAC
      - Gain stage
    - Error correction by adding redundancy
    - Digital calibration
    - Correction for inter-stage gain nonlinearity

# ADC Architectures

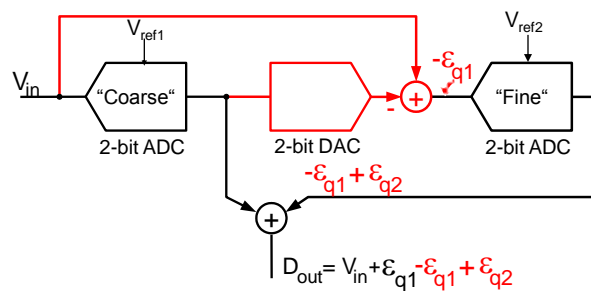
- Slope type converters
- Successive approximation
- Flash
- Interpolating & Folding
- ➔ • Residue type ADCs
  - Two-step Flash
  - Pipelined ADCs
  - ...
- Time-interleaved / parallel converter
- Oversampled ADCs

## Two-Step Example: (2+2)Bits



- Using only one ADC: output contains large quantization error
- "Missing voltage" or "residue" ( $-\epsilon_{q1}$ )
- Idea: Use second ADC to quantize and add  $-\epsilon_{q1}$

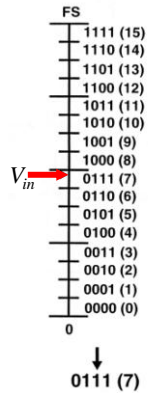
## Two Stage Example



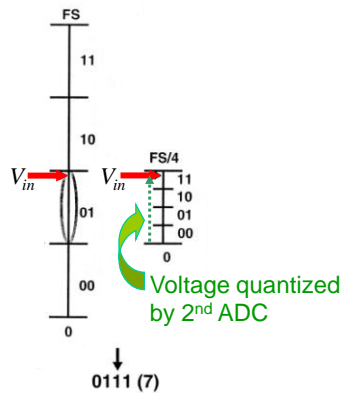
- Use DAC to compute missing voltage
- Add quantized representation of missing voltage
- Why does this help? How about  $\epsilon_{q2}$ ?
- Since maximum voltage at input of the 2<sup>nd</sup> ADC is  $V_{ref1}/4$  then for 2<sup>nd</sup> ADC  $V_{ref2} = V_{ref1}/4$  and thus  $\epsilon_{q2} = \epsilon_{q1}/4 = V_{ref1}/16 \rightarrow$  4bit overall resolution

# Two Step (2+2) Flash ADC

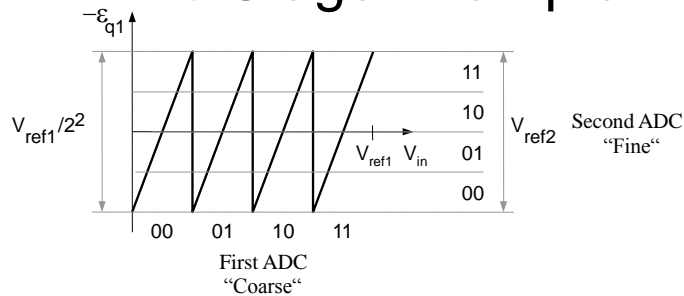
4-bit Straight Flash ADC



Ideal 2-step Flash ADC



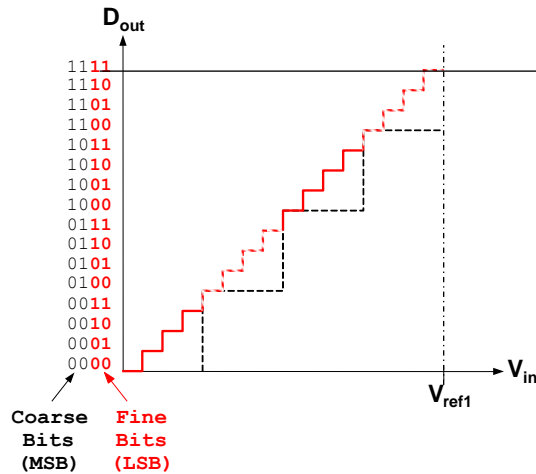
# Two Stage Example



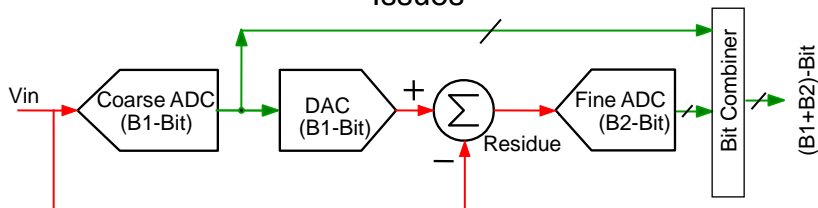
- Fine ADC is re-used  $2^2$  times
- Fine ADC's full scale range needs to span only 1 LSB of coarse quantizer

$$\epsilon_{q2} = \frac{V_{ref2}}{2^2} = \frac{V_{ref1}}{2^2 \cdot 2^2}$$

## Two-Stage (2+2) ADC Transfer Function

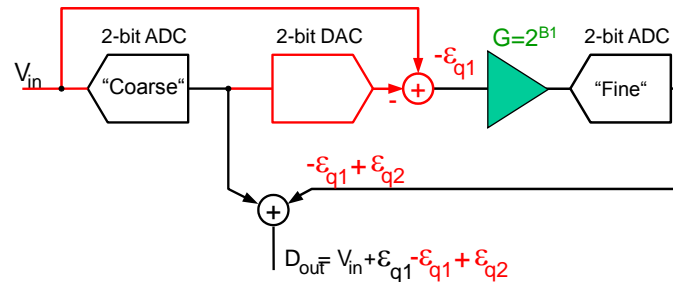


## Residue or Multi-Step Type ADC Issues



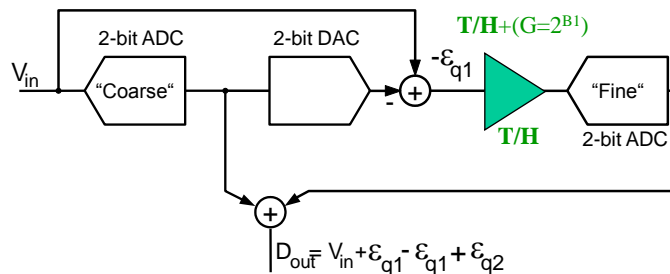
- Operation:
  - Coarse ADC determines MSBs
  - DAC converts the coarse ADC output to analog- Residue is found by subtracting ( $V_{in}-V_{DAC}$ )
  - Fine ADC converts the residue and determines the LSBs
  - Bits are combined in digital domain
- Issue:
  1. Fine ADC has to have  $FS=FS_{coarse}/2^{B1}$  & precision in the order of overall ADC  $1/2LSB$
  2. Speed penalty  $\rightarrow$  Need at least 1 clock cycle per extra series stage to resolve one sample

## Solution to Issue (1) Reducing Precision Required for Fine ADC



- Accuracy needed for fine ADC relaxed by introducing inter-stage gain
  - Example: By adding gain of  $x(G=2^{B1}=4)$  prior to fine ADC in (2+2)bit case, precision required for fine ADC is reduced to 2-bit only!
  - Additional advantage- coarse and fine ADC can be identical stages

## Solution to Issue (2) Increasing ADC Throughput

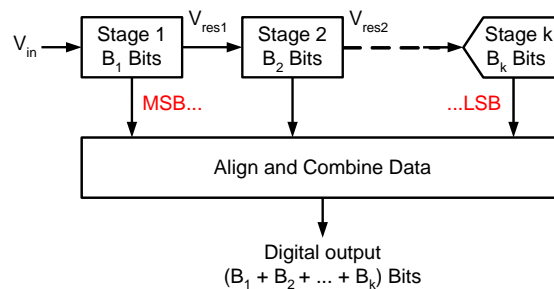


- Conversion time significantly decreased by employing T/H between stages
  - All stages busy at all times  $\rightarrow$  operation concurrent
  - During one clock cycle coarse & fine ADCs operate concurrently:
    - First stage samples/converts/generates residue of input signal sample #  $n$
    - While 2<sup>nd</sup> stage samples/converts residue associated with sample #  $n-1$

# Residue Type ADCs

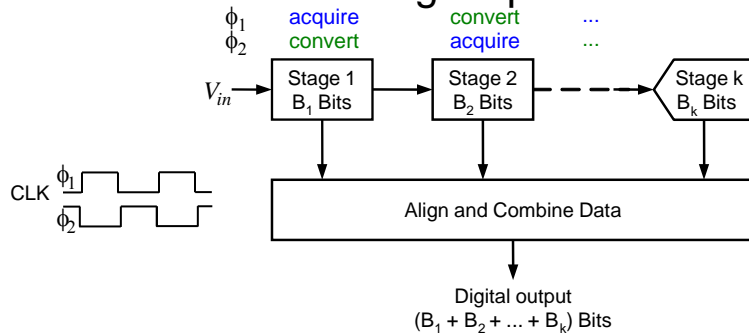
- Two-Step flash
- ➔ • Pipelined ADCs
  - Basic operation
  - Effect of sub-ADC, sub-DAC, gain stage non-idealities on overall ADC performance
    - Error correction by adding redundancy
    - Digital calibration
    - Correction for inter-stage gain nonlinearity
  - Implementation
    - Practical circuits
    - Stage scaling
    - Combining the bits
    - Stage implementation
      - Circuits
      - Noise budgeting
    - How many bits per stage?

## Pipeline ADC Block Diagram



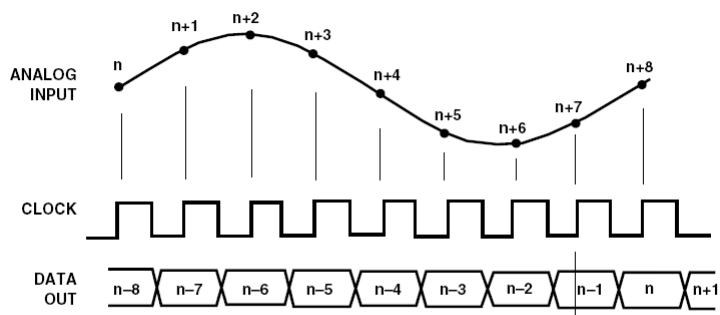
- Idea: Cascade several low resolution stages to obtain high overall resolution (e.g. 10bit ADC can be built with series of 10 ADCs each 1-bit only!)
- Each stage performs coarse A/D conversion and computes its quantization error, or "residue"
- All stages operate concurrently

## Pipeline ADC Concurrent Stage Operation



- Stages operate on the input signal like a shift register
- New output data **every** clock cycle, but each stage introduces at least  $\frac{1}{2}$  clock cycle latency

## Pipeline ADC Latency



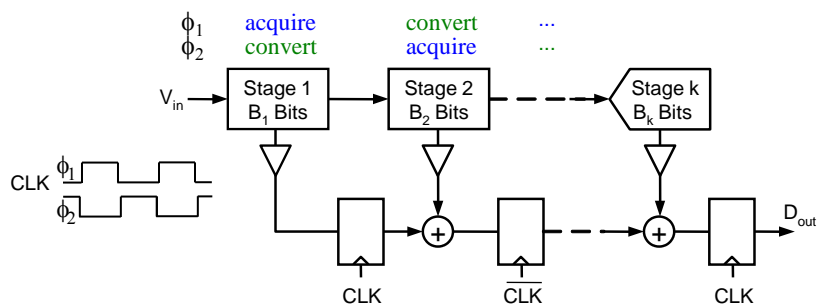
Note: One conversion per clock cycle & 8 clock cycle latency

[Analog Devices, AD 9226 12bit ADC Data Sheet]

## Pipelined ADC Characteristics

- Number of components (stages) grows linearly with resolution
- Pipelining
  - Trading latency for overall component count
  - Latency may be an issue in e.g. control systems
  - Throughput limited by speed of one stage → Fast
- Versatile: 8...16bits, 1...400MS/s
- One important feature of pipelined ADCs: many analog circuit non-idealities can be corrected digitally

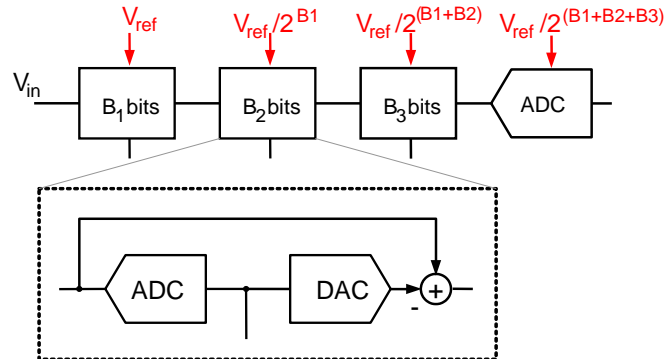
## Pipeline ADC Digital Data Alignment



- Digital shift register aligns sub-conversion results in time

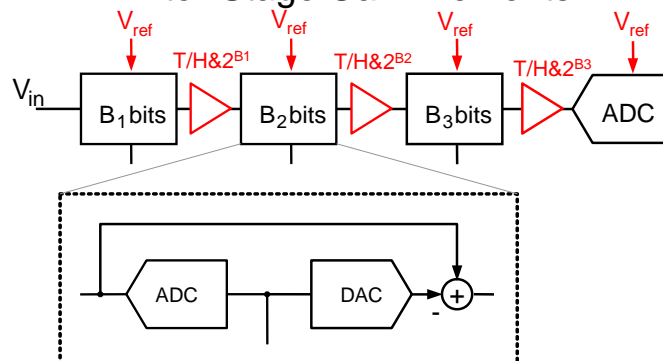


## Cascading More Stages



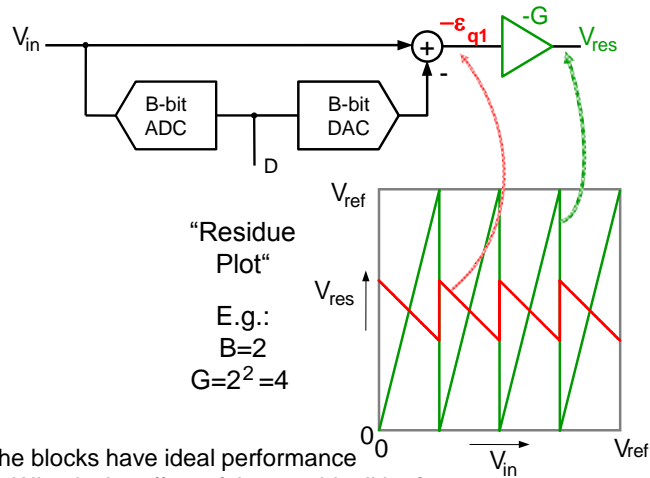
- LSB of last stage becomes very small
- All stages need to have full precision
- Impractical to generate several  $V_{ref}$

## Pipeline ADC Inter-Stage Gain Elements



- Practical pipelines by adding inter-stage gain  $\rightarrow$  use single  $V_{ref}$
- Precision requirements decrease down the pipe
  - Advantageous for noise, matching (later), power dissipation
- All stages can operate concurrently  $\rightarrow$  Throughput 1sample/clock cycle

## Complete Pipeline Stage



Note: None of the blocks have ideal performance

Question: What is the effect of the non-idealities?

## Pipeline ADC Errors

- Non-idealities associated with sub-ADCs, sub-DACs and gain stages  $\rightarrow$  error in overall pipeline ADC performance
- Need to find means to tolerate/correct errors
- Important sources of error
  - Sub-ADC errors- comparator offset
  - Gain stage offset
  - Gain stage gain error
  - Sub-DAC error



## Pipeline ADC Model

- If the "Analog" and "Digital" gain/loss is precisely matched:

$$D_{out} = V_{in,ADC} + \frac{\epsilon_{qn}}{\prod_{j=1}^{n-1} G_j} \quad \text{where } \epsilon_{qn} = \frac{V_{ref}}{2^{B_n}} \quad \& \ B_n = \# \text{ of bits in final stage}$$

$$D.R. = 20 \log \frac{rms \ FS \ Signal}{rms \ Quant. \ Noise} = 20 \log \frac{\frac{V_{ref}}{2\sqrt{2}}}{\frac{\sqrt{12} \times 2^{B_n}}{\prod_{j=1}^{n-1} G_j}} = 20 \log \left( \sqrt{\frac{3}{2}} \times 2^{B_n} \times \prod_{j=1}^{n-1} G_j \right)$$

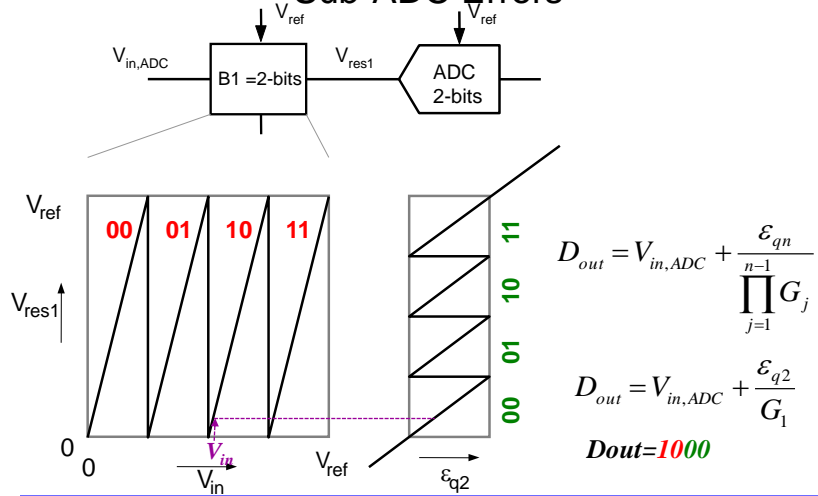
$$B_{ADC} \approx \log_2 \left( 2^{B_n} \times \prod_{j=1}^{n-1} G_j \right)$$

$$B_{ADC} \approx B_n + \log_2 \prod_{j=1}^{n-1} G_j$$

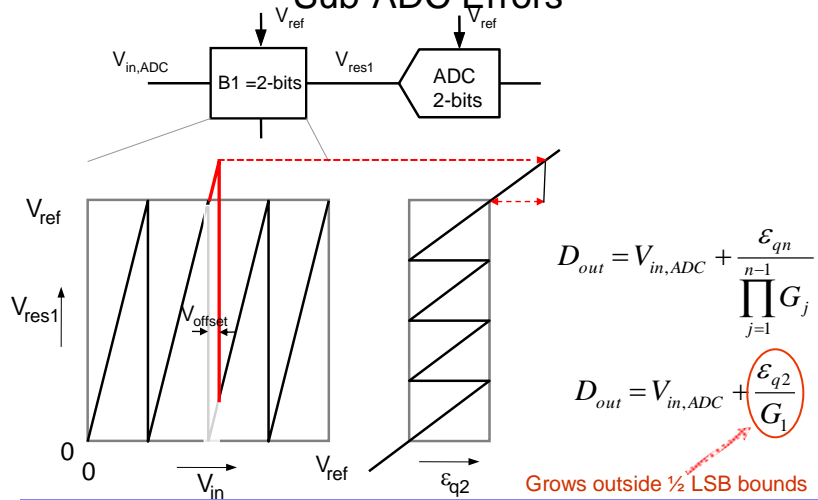
## Pipeline ADC Observations

- The aggregate **ADC resolution is independent of sub-ADC resolution!**
- *Effective* stage resolution  $B_j = \log_2(G_j)$
- **Overall conversion error does not (directly) depend on sub-ADC errors!**
- Only error term in  $D_{out}$  contains quantization error associated with the last stage
- So why do we care about sub-ADC errors?
  - Go back to two stage example

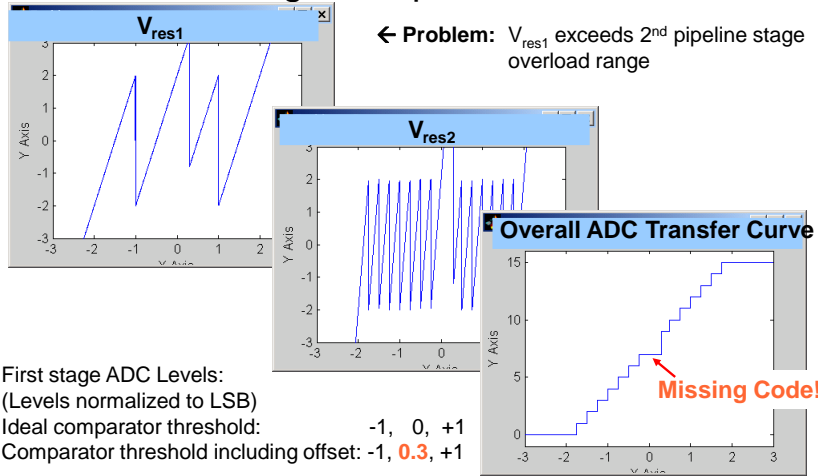
## Pipeline ADC Sub-ADC Errors



## Pipeline ADC Sub-ADC Errors



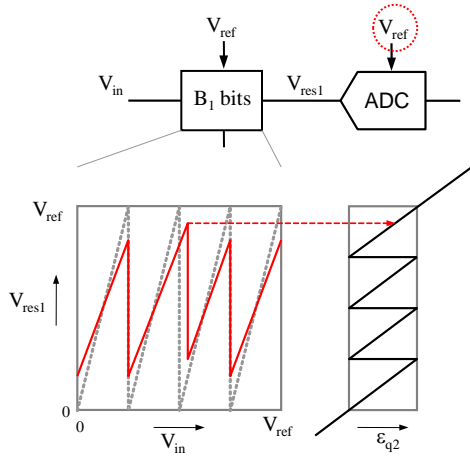
## Pipeline ADC 1<sup>st</sup>-Stage Comparator Offset



## Pipeline ADC Three Ways to Deal with Sub-ADC Errors

- All involve “sub-ADC redundancy”
- Redundancy in stage that produces errors
  - Choose gain for residue to be processed by the 2<sup>nd</sup> stage  $< 2^{B_1}$
  - Higher resolution sub-ADC & sub-DAC
- Redundancy in succeeding stage(s)

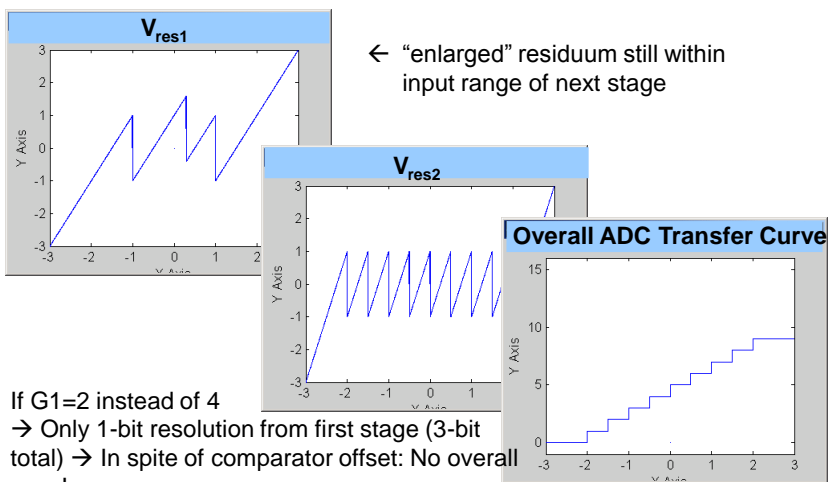
## (1) Inter-Stage Gain Following 1<sup>st</sup> Stage $< 2^{B_1}$



- Choose  $G_1$  less than  $2^{B_1}$
- Effective stage resolution could become non-integer  
 $B_{1eff} = \log_2 G_1$
- E.g. if  $G_1 = 3.8 \rightarrow B_{1eff} = 1.8\text{bit}$

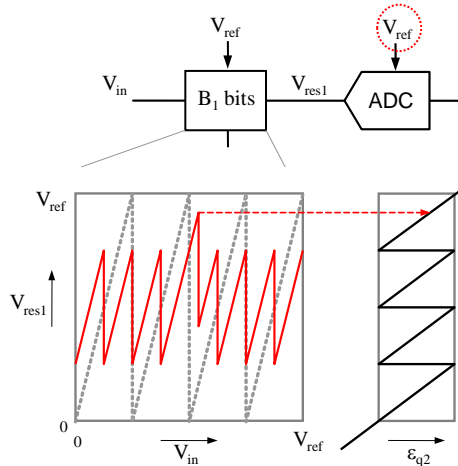
-Ref: A. Karanicolas et. al.,  
JSSC 12/1993

## Correction Through Redundancy



If  $G_1=2$  instead of 4  
 $\rightarrow$  Only 1-bit resolution from first stage (3-bit total)  $\rightarrow$  In spite of comparator offset: No overall error!

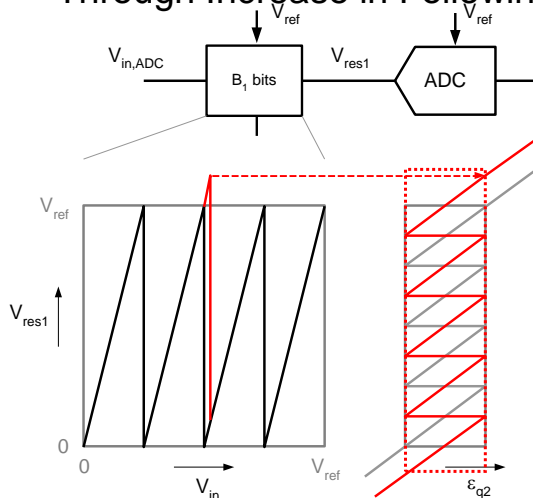
## (2) Higher Resolution Sub-ADC



- Keep  $G_1$  precise power of two (e.g.  $G_1=4$ )
- Add extra decision levels in sub-ADC (e.g. add 1 extra bit to 1<sup>st</sup> stage)
- E.g.  $B_1=B_{1\text{eff}}+1$

-Ref: Singer et. al., VSLI 1996

## (3) Over-Range Accommodation Through Increase in Following Stage Resolution



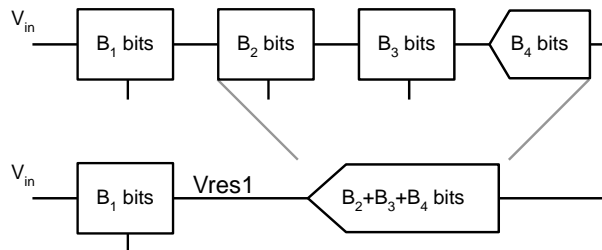
- No redundancy in stage with errors
- Add extra decision levels in succeeding stage

❖ Ref: Opris et. al., JSSC 12/1998



## Redundancy

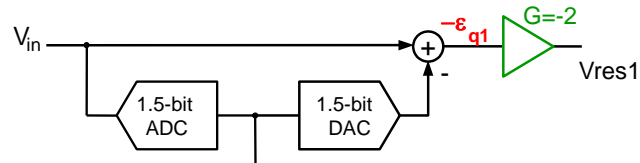
- The preceding analysis applies to any stage in an n-stage pipeline
- Can always perceive a multi-stage pipelined ADC as a single stage + backend ADC



## Redundancy

- In literature, sub-ADC redundancy schemes are often called "digital correction" – a misnomer!
- No error correction takes place
- We can **tolerate** sub-ADC errors as long as:
  - The residues stay "within the box", or
  - Another stage downstream "returns the residue to "within the box" before it reaches last quantizer
- Let's calculate tolerable errors for popular "1.5 bits/stage" topology

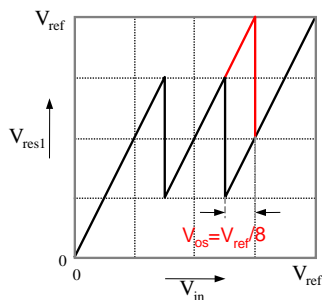
## 1.5-Bit/Stage Pipelined ADC



- $G=-2$
- Effective bit/stage  $\rightarrow B_{\text{eff}}=\log_2|G|=\log_2 2=1$
- Actual bit/stage  $\rightarrow B=\log_2(2+1)=1.589\dots$
- 1bit/stage + 0.5bit  $\rightarrow$  redundancy

❖ Ref: Lewis et. al., JSSC 3/1992

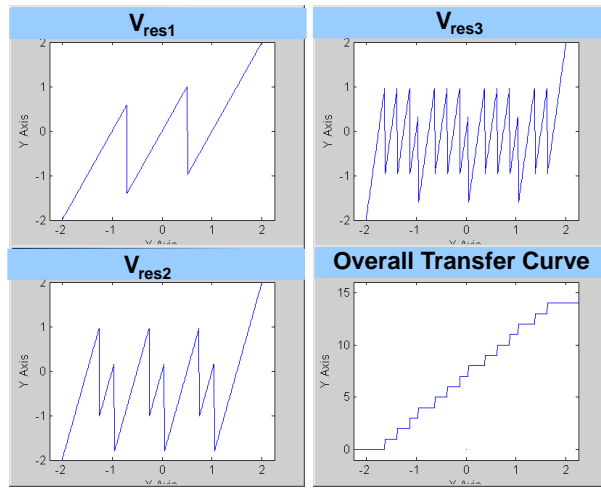
## 1.5 Bits/Stage Example



- Comparators threshold levels placed strategically
- $G=2$
- $B_{\text{eff}}=\log_2 G=\log_2 2=1$
- $B=\log_2(2+1)=1.589\dots$
- 0.5bit  $\rightarrow$  redundancy

❖ Ref: Lewis et. al., JSSC 3/1992

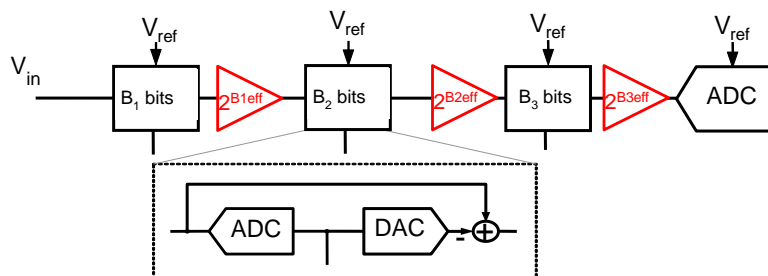
## 3-Stage 1.5-bit-per-Stage Pipelined ADC



- All three stages  
→ Comparator with offset
- Overall transfer curve
  - No missing codes
  - Some DNL error

Ref: S. Lewis et al. "A 10-b 20-MS/s Analog-to-Digital Converter," J. Solid-State Circ., pp. 351-8, March 1992

## Summary So Far Pipelined A/D Converters

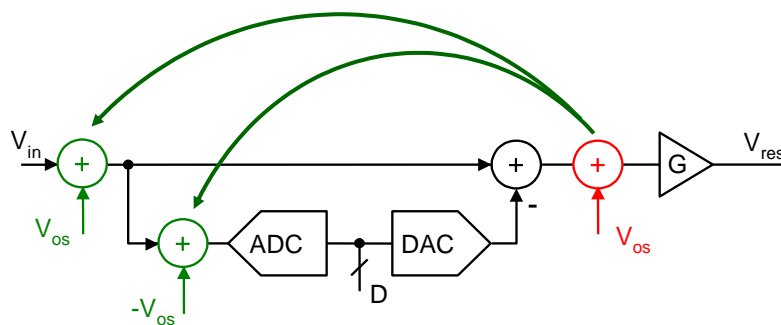


- Cascade of low resolution stages
  - Stages operate concurrently- trades latency for overall component count
  - Throughput limited by speed of one stage → Fast
- Errors and correction
  - Built-in redundancy compensate for sub-ADC inaccuracies (interstage gain:  $G=2^{B_{\text{neff}}}$ ,  $B_{\text{neff}} < B_n$ )

# Pipeline ADC Errors

- Non-idealities associated with sub-ADCs, sub-DACs and gain stages → error in overall pipeline ADC performance
- Need to find means to tolerate/correct errors
- Important sources of error
  - Sub-ADC errors- comparator offset
  - – Gain stage offset
  - Gain stage error
  - Sub-DAC error

## Inter-Stage Amplifier Offset

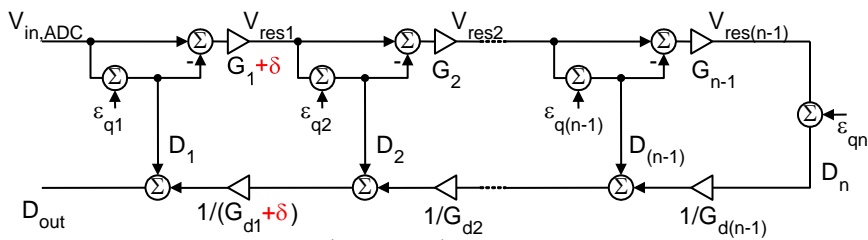


- Input referred converter offset – usually no problem
- Equivalent sub-ADC offset - accommodated through adequate redundancy

# Pipeline ADC Errors

- Non-idealities associated with sub-ADCs, sub-DACs and gain stages → error in overall pipeline ADC performance
- Need to find means to tolerate/correct errors
- Important sources of error
  - Sub-ADC errors- comparator offset
  - Gain stage offset
  - Gain stage gain error
  - Sub-DAC error

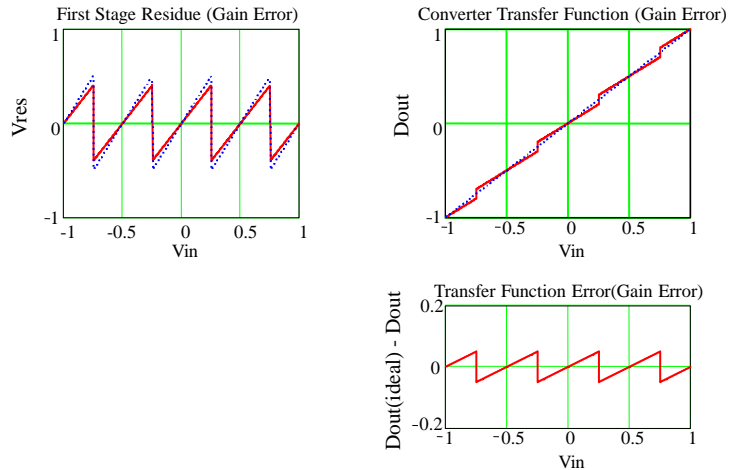
## Gain Stage Gain Error



$$D_{out} = V_{in,ADC} + \epsilon_{q1} \left( 1 - \frac{G_1 + \delta}{G_{d1} + \delta} \right) + \frac{\epsilon_{q2}}{G_{d1}} \left( 1 - \frac{G_2}{G_{d2}} \right) + \dots + \frac{\epsilon_{q(n-1)}}{\prod_{j=1}^{n-2} G_{dj}} \left( 1 - \frac{G_{(n-1)}}{G_{d(n-1)}} \right) + \frac{\epsilon_{qn}}{\prod_{j=1}^{n-1} G_{dj}}$$

- Resolution is function of  $\log_2 G$  therefore error in  $G$  affects resolution
- Small amount of gain error can be tolerated

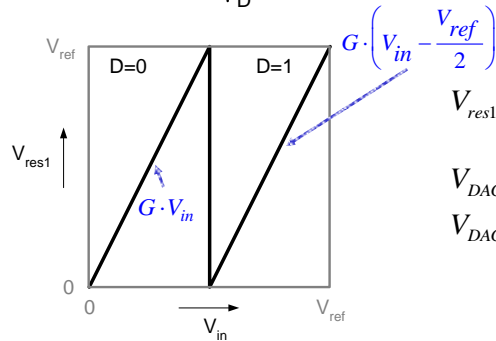
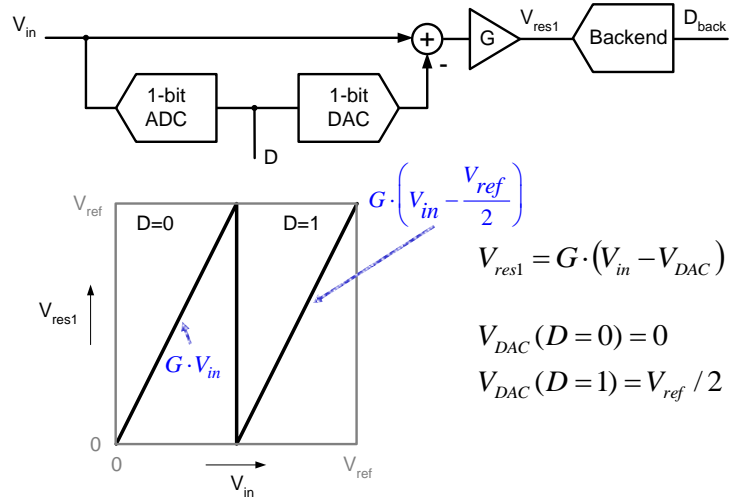
## Interstage Gain Error



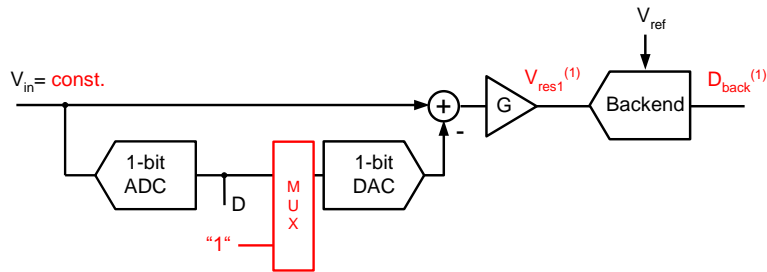
## Gain Stage Gain Inaccuracy

- Gain error can be compensated in digital domain – "Digital Calibration"
- Problem: Need to measure/calibrate digital correction coefficient
- Example: Calibrate 1-bit first stage
- Objective: Measure  $G$  in digital domain

## ADC Model



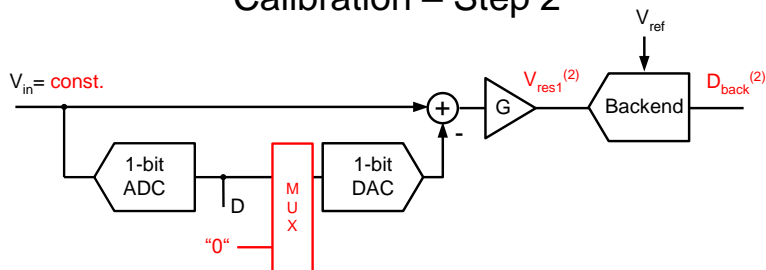
## Gain Stage Inaccuracy Calibration – Step 1



$$V_{res1}^{(1)} = G \cdot (V_{in} - V_{ref} / 2)$$

$$D_{back}^{(1)} = G \cdot \frac{(V_{in} - V_{ref} / 2)}{V_{ref}} \rightarrow \text{store}$$

## Gain Stage Inaccuracy Calibration – Step 2



$$V_{res1}^{(2)} = G \cdot (V_{in} - 0)$$

$$D_{back}^{(2)} = G \cdot \frac{(V_{in} - 0)}{V_{ref}} \rightarrow \text{store}$$

## Gain Stage Inaccuracy Calibration – Evaluate

$$D_{back}^{(1)} = G \cdot \frac{(V_{in} - V_{ref}/2)}{V_{ref}}$$

$$-D_{back}^{(2)} = G \cdot \frac{(V_{in} - 0)}{V_{ref}}$$

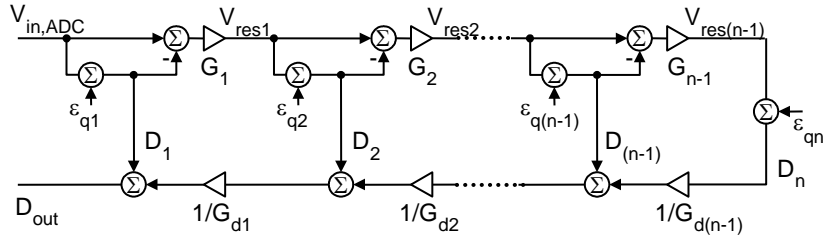
---


$$D_{back}^{(1)} - D_{back}^{(2)} = \frac{1}{2} \cdot G$$

- To minimize the effect of backend ADC noise → perform measurement several times and take the average



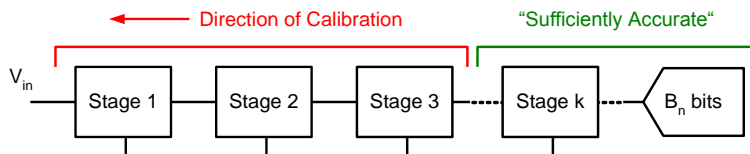
## Accuracy Bootstrapping



$$D_{out} = V_{in,ADC} + \varepsilon_{q1} \left( 1 - \frac{G_1}{G_{d1}} \right) + \frac{\varepsilon_{q2}}{G_{d1}} \left( 1 - \frac{G_2}{G_{d2}} \right) + \dots + \frac{\varepsilon_{q(n-1)}}{\prod_{j=1}^{n-2} G_{dj}} \left( 1 - \frac{G_{(n-1)}}{G_{d(n-1)}} \right) + \frac{\varepsilon_{qn}}{\prod_{j=1}^{n-1} G_{dj}}$$

- Highest sensitivity to gain errors in front-end stages

## "Accuracy Bootstrapping"




Ref:

A. N. Karanicolas et al. "A 15-b 1-Msample/s digitally self-calibrated pipeline ADC," *IEEE J. of Solid-State Circuits*, pp. 1207-15, Dec. 1993

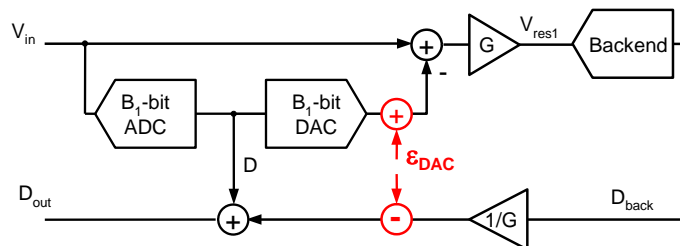
E. G. Soenen et al., "An architecture and an algorithm for fully digital correction of monolithic pipelined ADCs," *TCAS II*, pp. 143-153, March 1995

L. Singer et al., "A 12 b 65 MSample/s CMOS ADC with 82 dB SFDR at 120 MHz," *ISSCC 2000, Digest of Tech. Papers.*, pp. 38-9 (calibration in opposite direction!)

# Pipeline ADC Errors

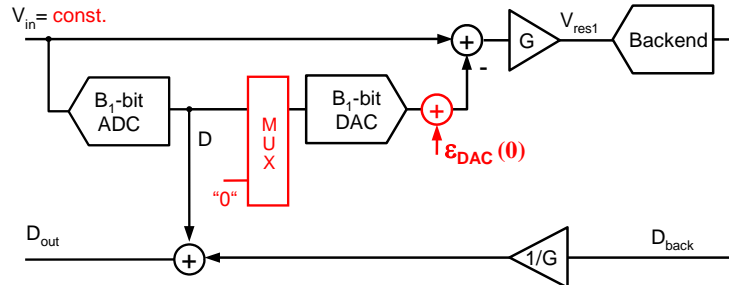
- Non-idealities associated with sub-ADCs, sub-DACs and gain stages  $\rightarrow$  error in overall pipeline ADC performance
- Need to find means to tolerate/correct errors
- Important sources of error
  - Sub-ADC errors- comparator offset
  - Gain stage offset
  - Gain stage error
  -  – Sub-DAC error

# DAC Errors



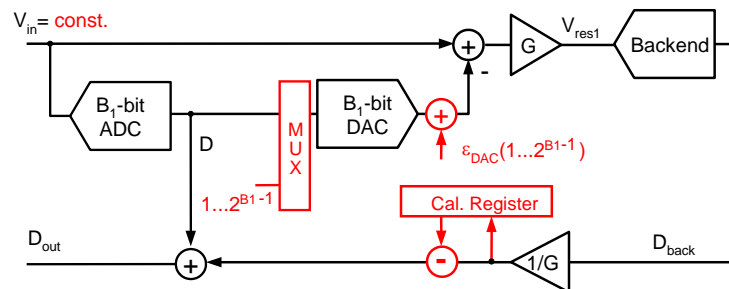
- Can be corrected digitally as well
- Same calibration concept as gain errors
  - $\rightarrow$  Vary DAC codes & measure errors via backend ADC

# DAC Calibration – Step 1



- $\epsilon_{DAC}(0)$  equivalent to offset - ignore

# DAC Calibration – Step $2 \dots 2^{B1}$



- Stepping through DAC codes  $1 \dots 2^{B1} - 1$  yields all incremental correction values
- Measurements repeated and averages to account for variance associated with noise

