



# **L12: Analog Building Blocks**

## **(OpAmps, A/D, D/A)**



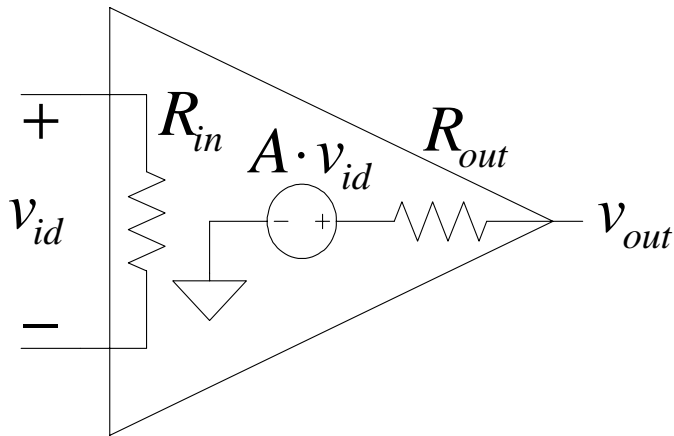
**Acknowledgement: Dave Wentzloff**



# Introduction to Operational Amplifiers



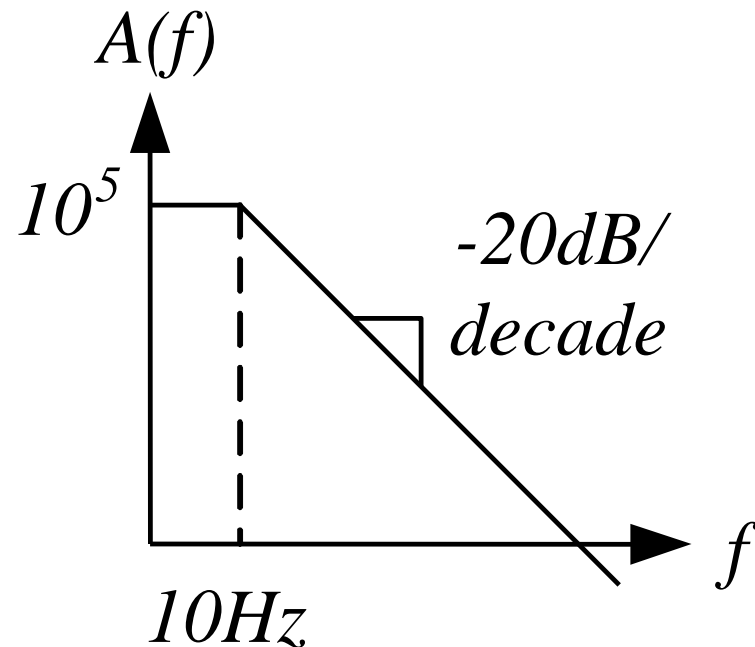
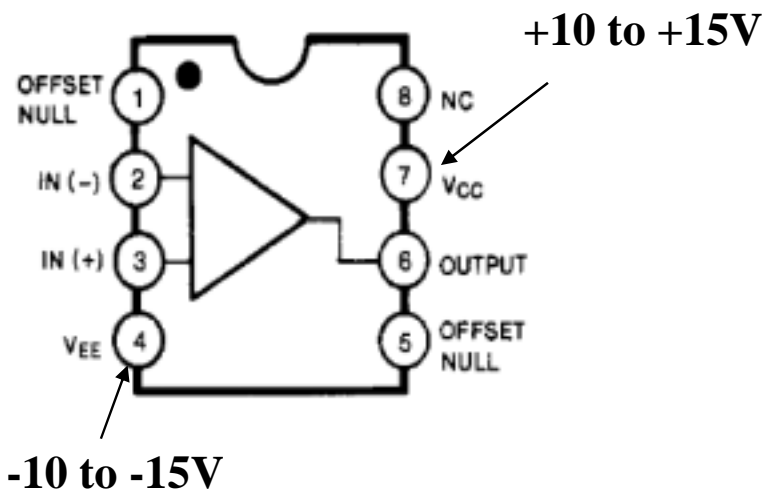
## DC Model



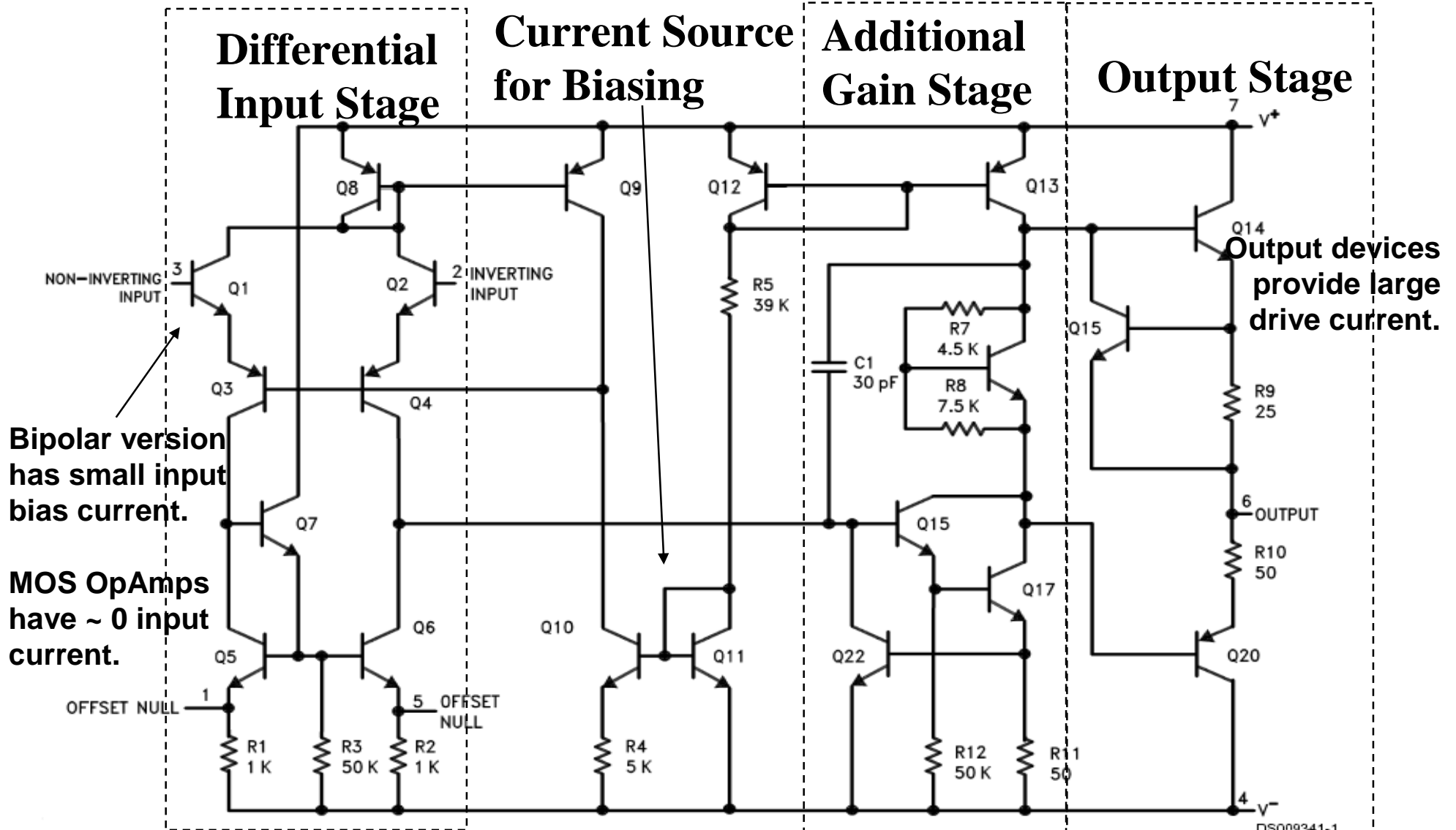
- Typically very high input resistance  $\sim 300\text{K}\Omega$
- High DC gain ( $\sim 10^5$ )
- Output resistance  $\sim 75\Omega$

$$V_{out} = A(f) \cdot V_{in}$$

## LM741 Pinout



# The Inside of a 741 OpAmp



Bipolar version has small input bias current.

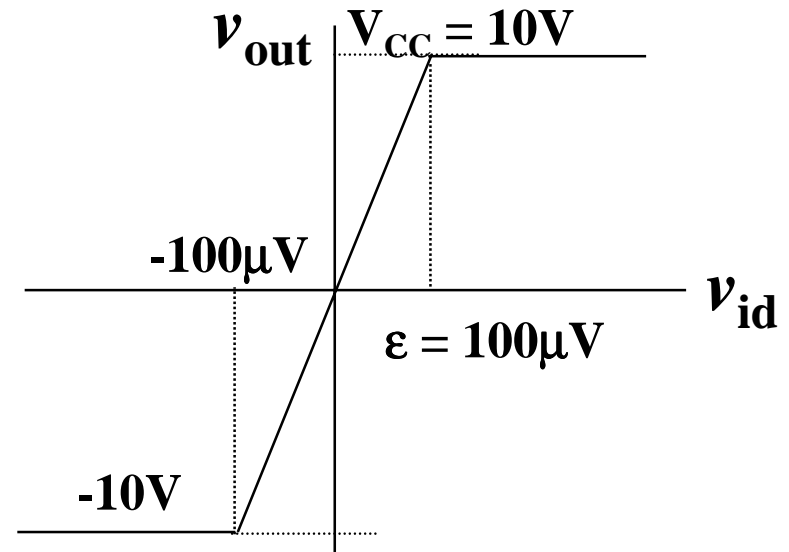
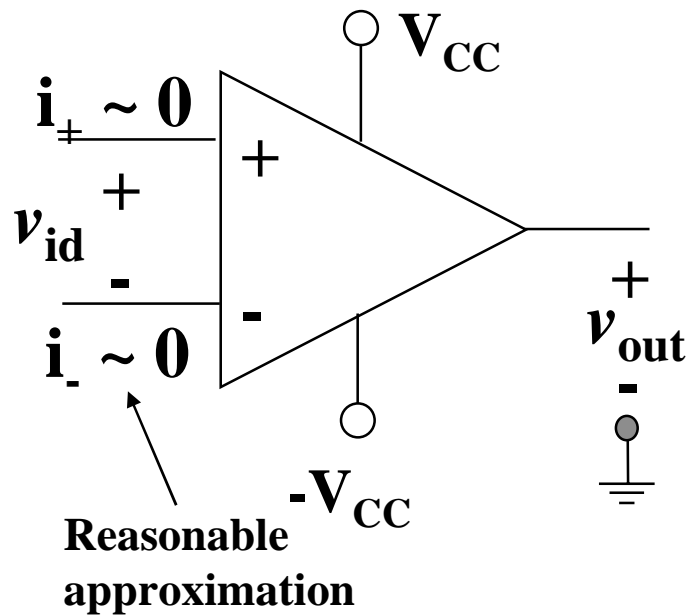
MOS OpAmps have ~ 0 input current.

Output devices provide large drive current.

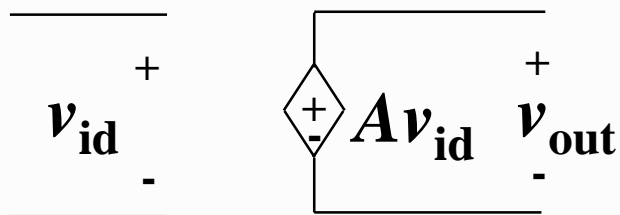
**Gain is Sensitive to Operating Condition (e.g., Device, Temperature, Power Supply Voltage, etc.).**



# Simple Model for an OpAmp

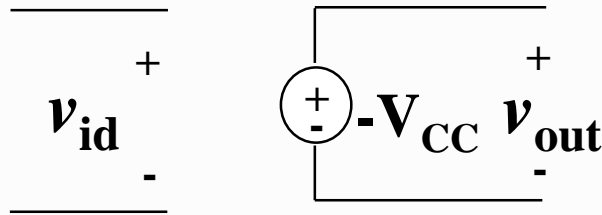


## Linear Mode



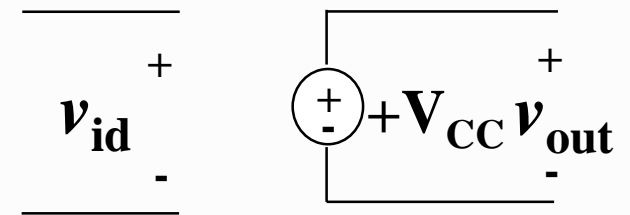
If-  $V_{CC} < v_{out} < V_{CC}$

## Negative Saturation



$v_{id} < -\epsilon$

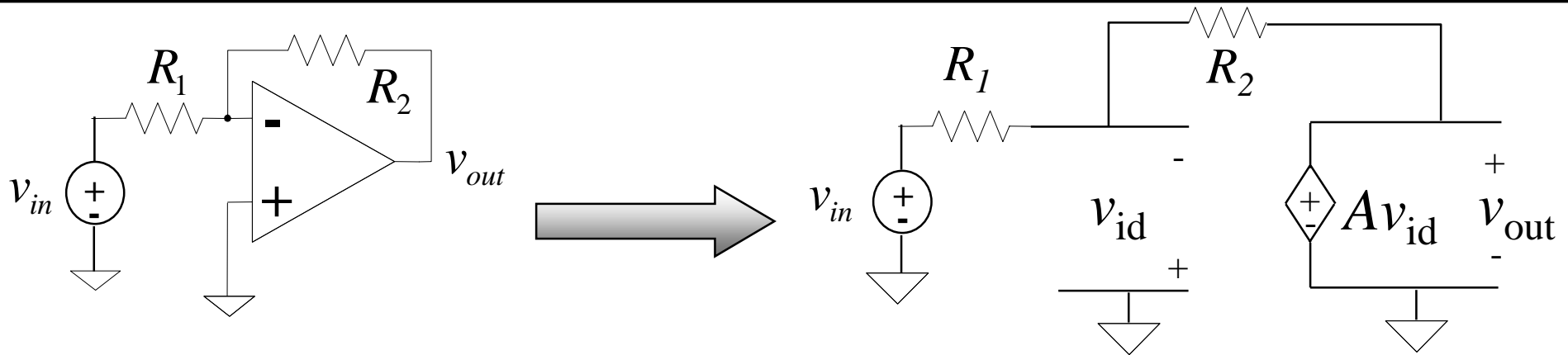
## Positive Saturation



$v_{id} > \epsilon$

Small input range for “Open” loop configuration

# The Power of Feedback



$$\frac{v_{in} + v_{id}}{R_1} + \frac{v_{out} + v_{id}}{R_2} = 0 \quad v_{id} = \frac{v_{out}}{A} \quad \frac{v_{in}}{R_1} = -\frac{v_{out}}{A} \left[ \frac{1}{R_1} + \frac{A}{R_2} + \frac{1}{R_2} \right]$$

$$v_{out} = -\frac{R_2 A}{(1 + A)R_1 + R_2} \approx -\frac{R_2}{R_1} \text{ (if } A \gg 1)$$

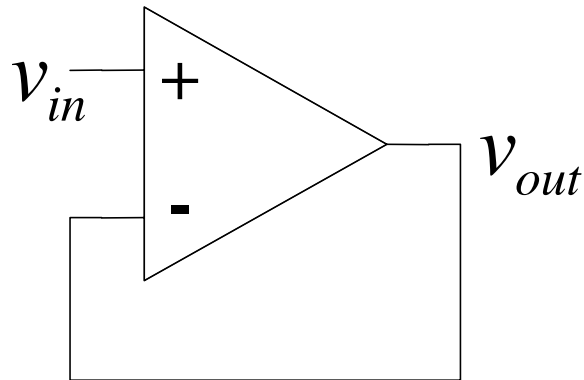
- Overall (closed loop) gain does not depend on open loop gain.
- Trade gain for robustness.
- Easier analysis approach: “virtual short circuit approach”
  - $v_+ = v_- = 0$  if OpAmp is linear



# Basic OpAmp Circuits

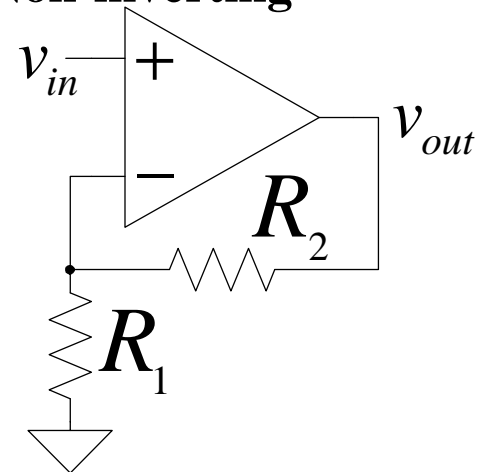


## Follower



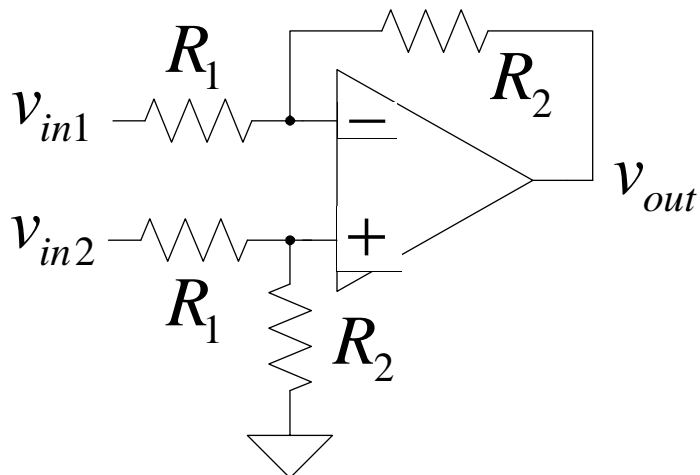
$$v_{out} \approx v_{in}$$

## Non-inverting



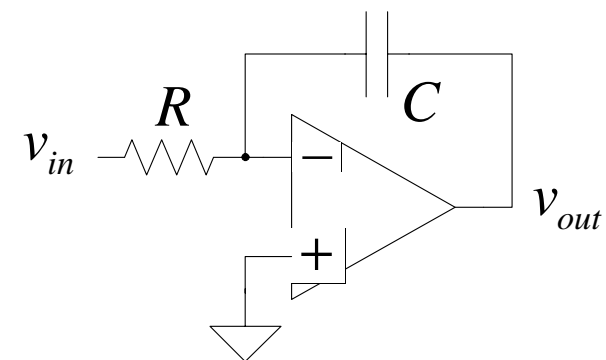
$$v_{out} \approx \frac{R_1 + R_2}{R_1} v_{in}$$

## Differential Input



$$v_{out} \approx \frac{R_2}{R_1} (v_{in2} - v_{in1})$$

## Integrator



$$v_{out} \approx -\frac{1}{RC} \int_{-\infty}^t v_{in} dt$$



# Use With Open Loop & Positive Feedback



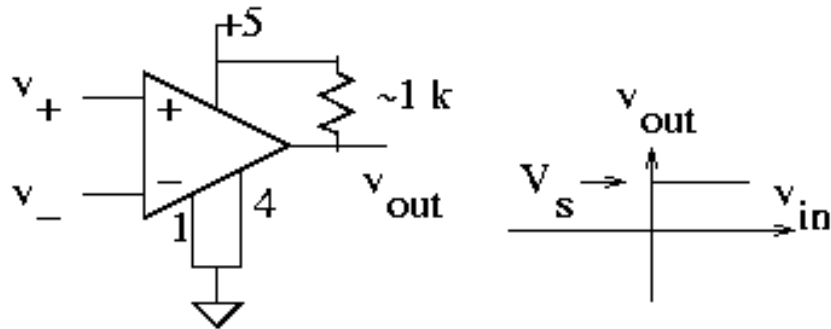
## Analog Comparator:

Is  $V_+ > V_-$  ?

The Output is a **DIGITAL** signal.

The external pull up resistor is often forgotten.

Analog Comparator: Analog to TTL  
LM 311 Needs Pull-Up



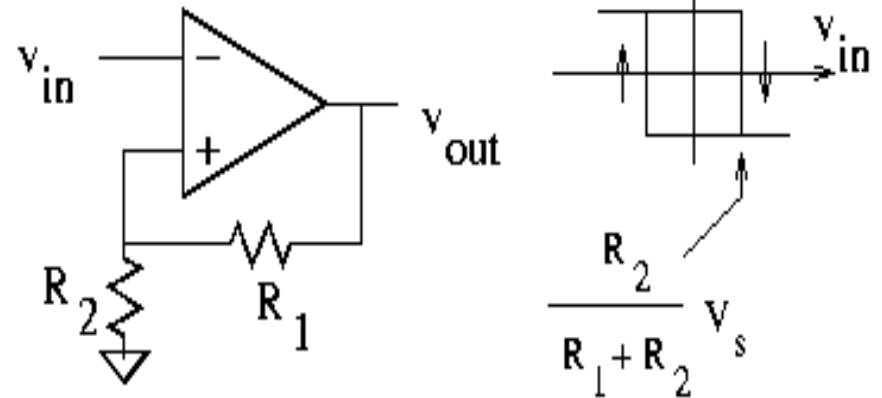
**LM311 is a single supply comparator.**

## Schmitt Trigger:

Squares up signals

Op Amp as Schmitt Trigger:

Output has hysteresis

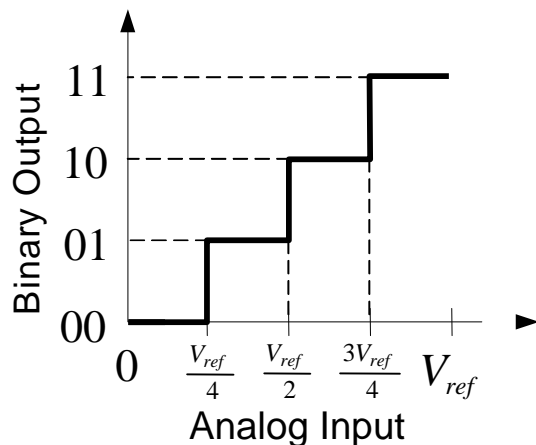




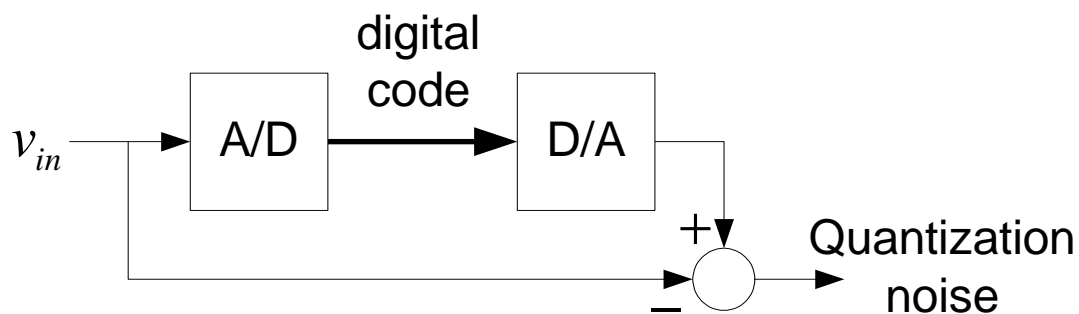
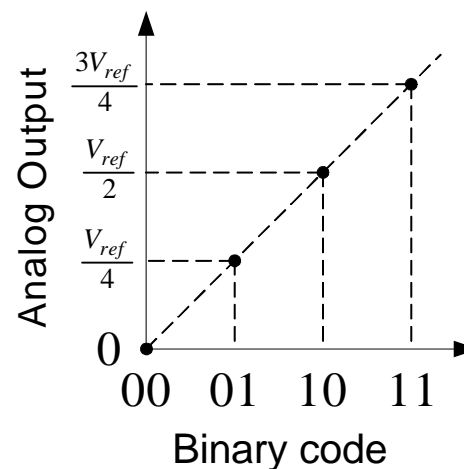
# Data Conversion: Quantization Noise



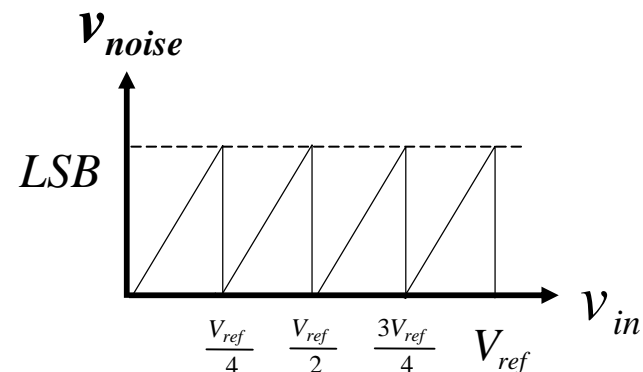
## A/D Conversion



## D/A Conversion



- Quantization noise exists even with *ideal* A/D and D/A converters.

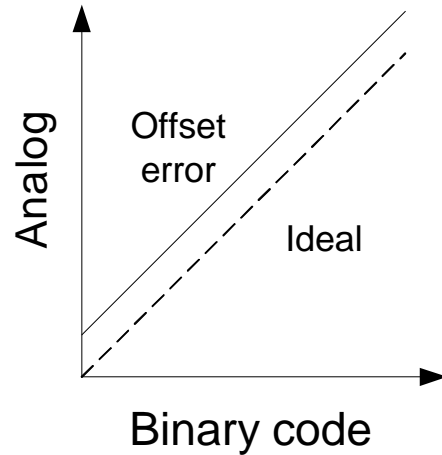




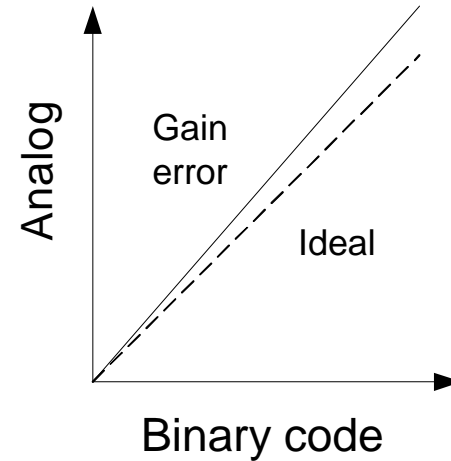
# Non idealities in Data Conversion



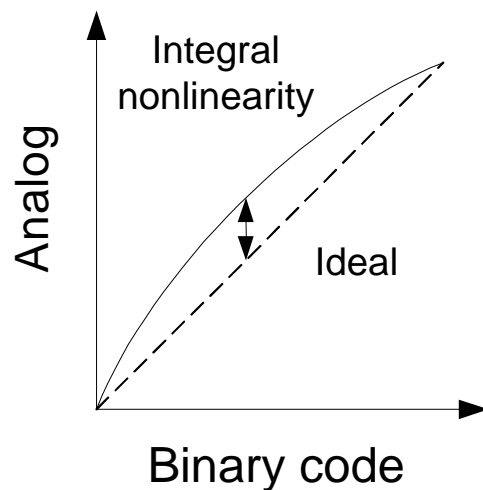
**Offset** – a constant voltage offset that appears at the output when the digital input is 0



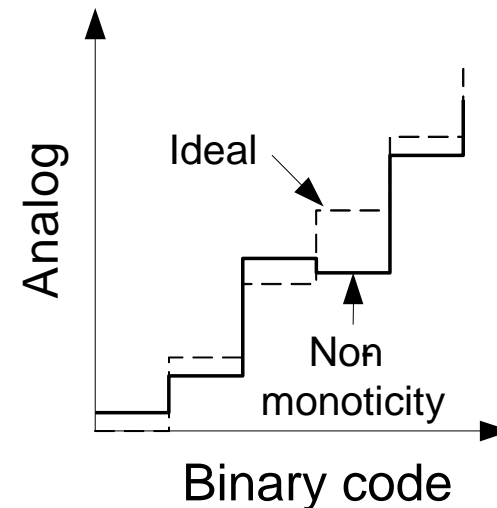
**Gain error** – deviation of slope from ideal value of 1



**Integral nonlinearity** – maximum deviation from the ideal analog output voltage

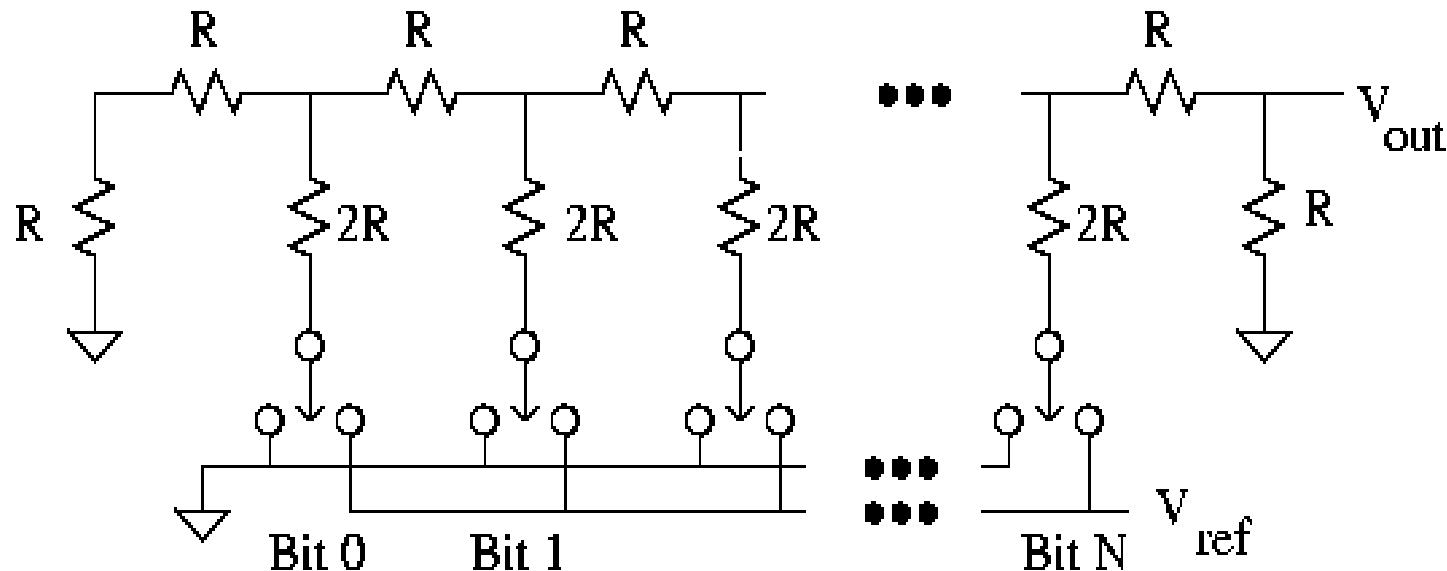


**Differential nonlinearity** – the largest increment in analog output for a 1 bit change





# R 2R Ladder DAC Architecture

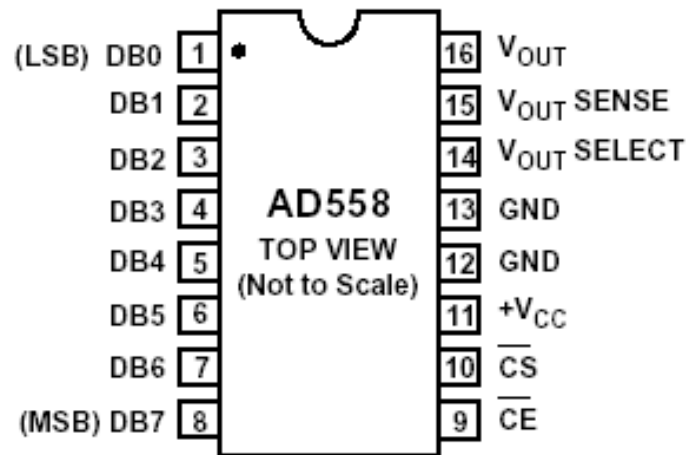


$$V_{out} = \frac{1}{6} V_{ref} \left[ B_7 + \frac{1}{2} B_6 + \frac{1}{4} B_5 + \dots + \frac{1}{128} B_0 \right]$$

- Note that the driving point impedance (resistance) is the same for each cell.
- R 2R Ladder achieves large current division ratios with only two resistor values.



# DAC (AD 558) Specs- Used in Lab 3



- 8 bit DAC
- Single Supply Operation: 5V to 15V
- Integrates required references (bandgap voltage reference)
- Uses a R/R resistor ladder
- Settling time 1μs
- Programmable output range from 0V to 2.56V or 0V to 10V
- Simple latch based interface

Input Logic Coding

Digital Input Code			Output Voltage	
Binary	Hexadecimal	Decimal	2.56 V Range	10.000 V Range
0000 0000	00	0	0	0
0000 0001	01	1	0.010 V	0.039 V
0000 0010	02	2	0.020 V	0.078 V
0000 1111	0F	15	0.150 V	0.586 V
0001 0000	10	16	0.160 V	0.625 V
0111 1111	7F	127	1.270 V	4.961 V
1000 0000	80	128	1.280 V	5.000 V
1100 0000	C0	192	1.920 V	7.500 V
1111 1111	FF	255	2.55 V	9.961 V



# Chip Architecture and Interface

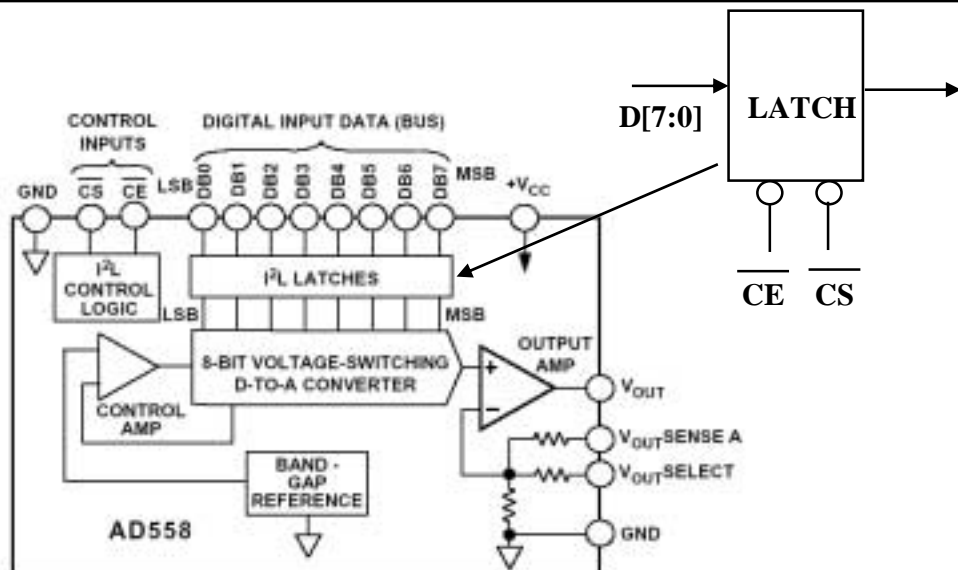


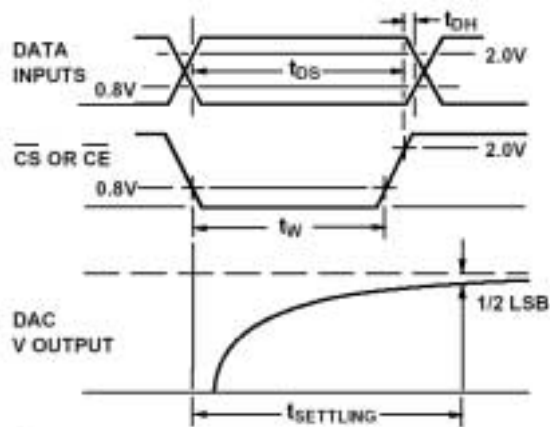
Table I. AD558 Control Logic Truth Table

Input Data	$\overline{CE}$	$\overline{CS}$	DAC Data	Latch Condition
0	0	0	0	"Transparent"
1	0	0	1	"Transparent"
0	g	0	0	Latching
1	g	0	1	Latching
0	0	g	0	Latching
1	0	g	1	Latching
X	1	X	Previous Data	Latched
X	X	1	Previous Data	Latched

NOTES

X = Does not matter.

g = Logic Threshold at Positive-Going Transition.



$t_W$  = STORAGE PULSE WIDTH = 200ns MIN

$t_{DH}$  = DATA HOLD TIME = 10ns MIN

$t_{DS}$  = DATA SETUP TIME = 200ns MIN

$t_{SETTLING}$  = DAC OUTPUT SETTTLING TIME TO  $\pm 1/2$  LSB

Outputs are noisy when input bits settle, so it is best to have inputs stable before latching the input data.

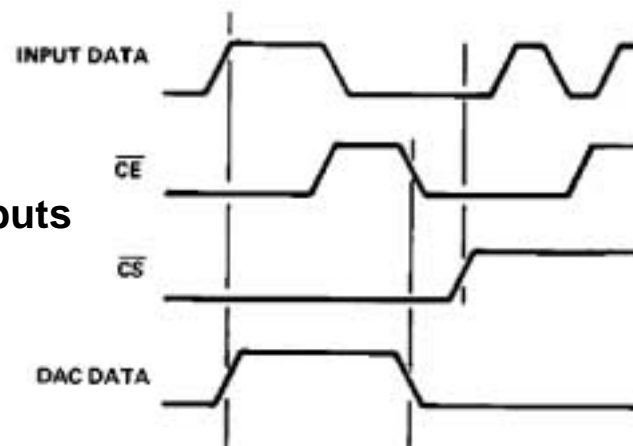
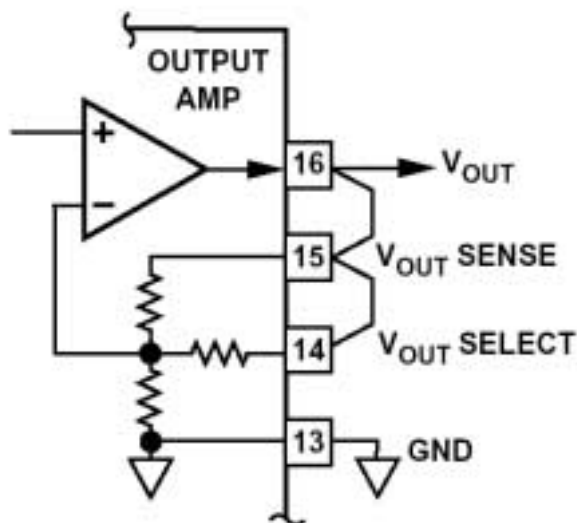


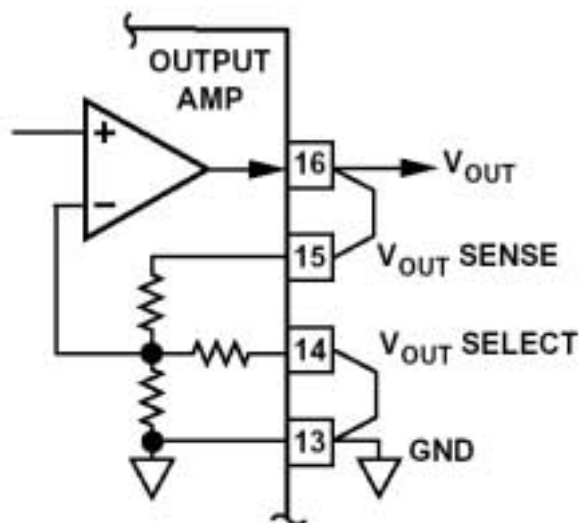
Figure 6. AD558 Control Logic Function



# Setting the Voltage Range



a. 0 V to 2.56 V Output Range



b. 0 V to 10 V Output Range

Very similar to a non-inverting amp

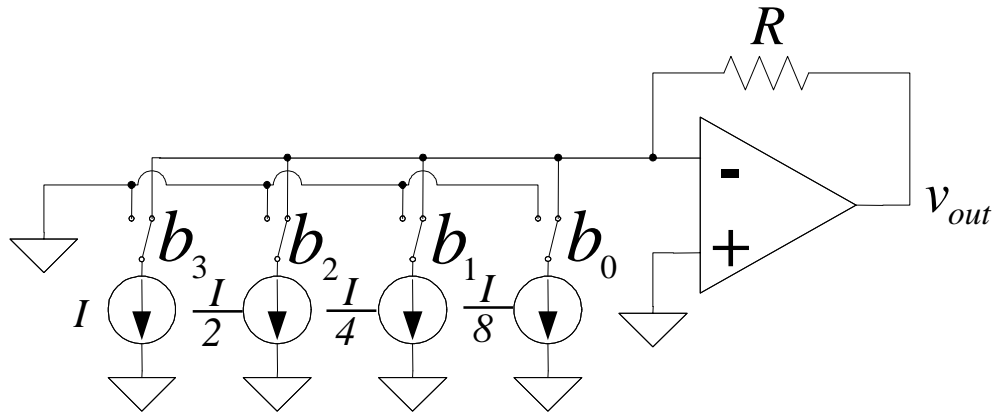
Strap output for different voltage ranges

Input Logic Coding

Digital Input Code			Output Voltage	
Binary	Hexadecimal	Decimal	2.56 V Range	10.000 V Range
0000 0000	00	0	0	0
0000 0001	01	1	0.010 V	0.039 V
0000 0010	02	2	0.020 V	0.078 V
0000 1111	0F	15	0.150 V	0.586 V
0001 0000	10	16	0.160 V	0.625 V
0111 1111	7F	127	1.270 V	4.961 V
1000 0000	80	128	1.280 V	5.000 V
1100 0000	C0	192	1.920 V	7.500 V
1111 1111	FF	255	2.55 V	9.961 V



# Another Approach: Binary Weighted DAC

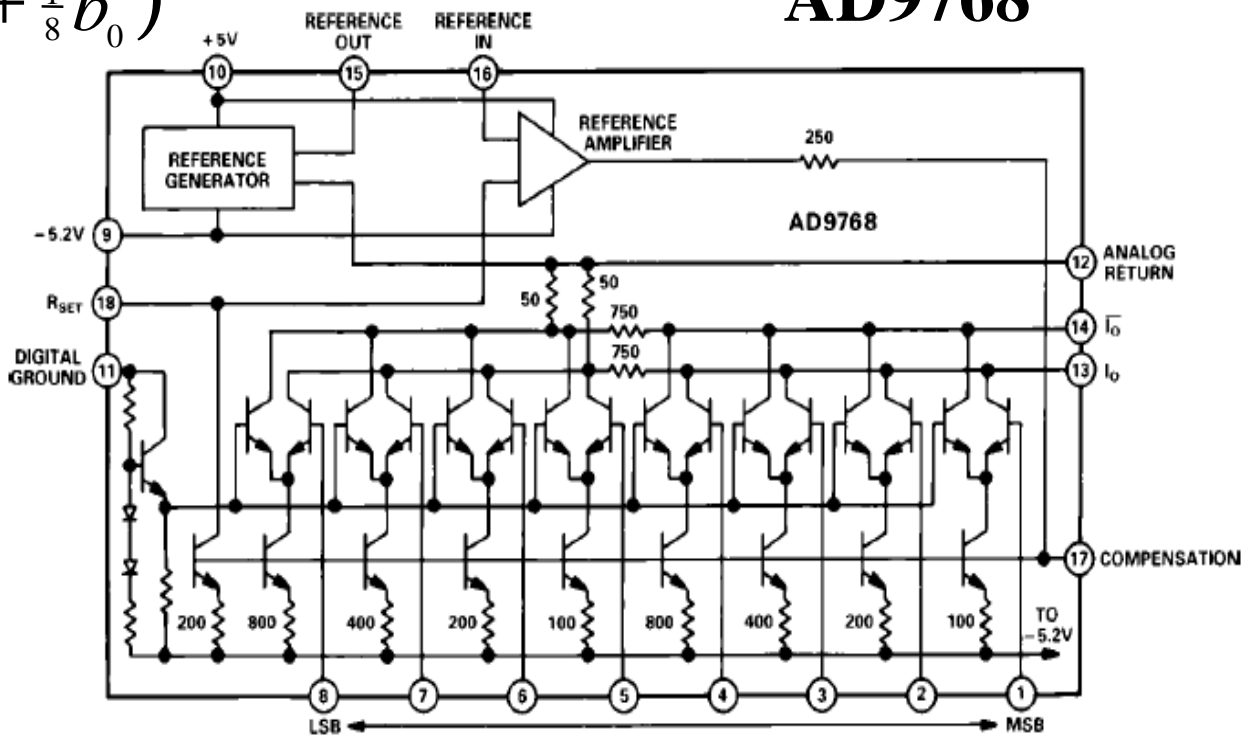


- Switch binary weighted currents
- MSB to LSB current ratio is  $2^N$ .

$$v_{out} = -IR \left( b_3 + \frac{1}{2}b_2 + \frac{1}{4}b_1 + \frac{1}{8}b_0 \right)$$

- Analog Devices AD9768 uses two banks of ratioed currents.
- Additional current division performed by  $750 \Omega$  resistor between the two banks.

## AD9768



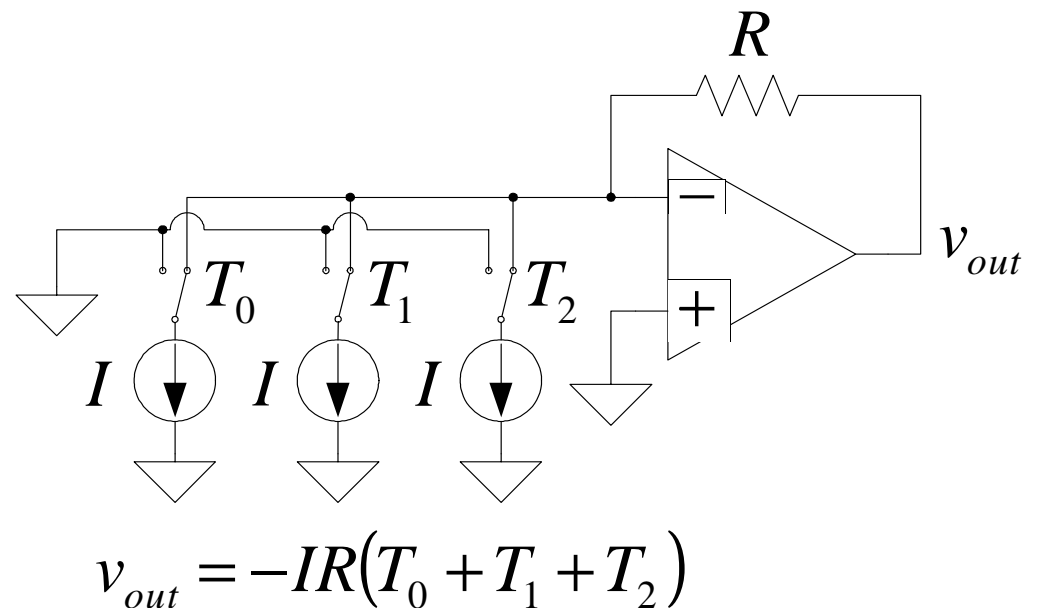
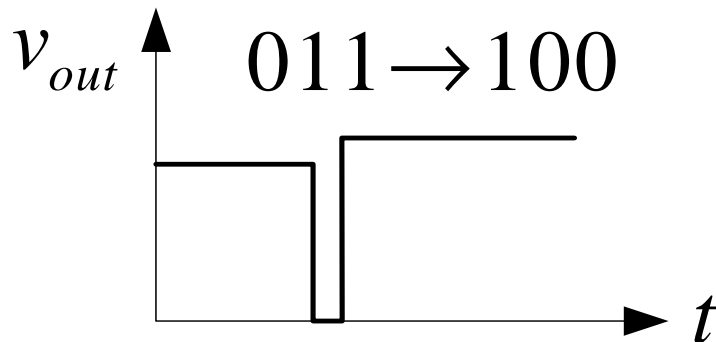


# Glitching and Thermometer D/A



- Glitching is caused when switching times in a D/A are not synchronized.
- Example: Output changes from 011 to 100 – MSB switch is delayed.
- Filtering reduces glitch but increases the D/A settling time.
- One solution is a thermometer code D/A – requires  $2^N - 1$  switches but no ratioed currents.

Binary		Thermometer		
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1

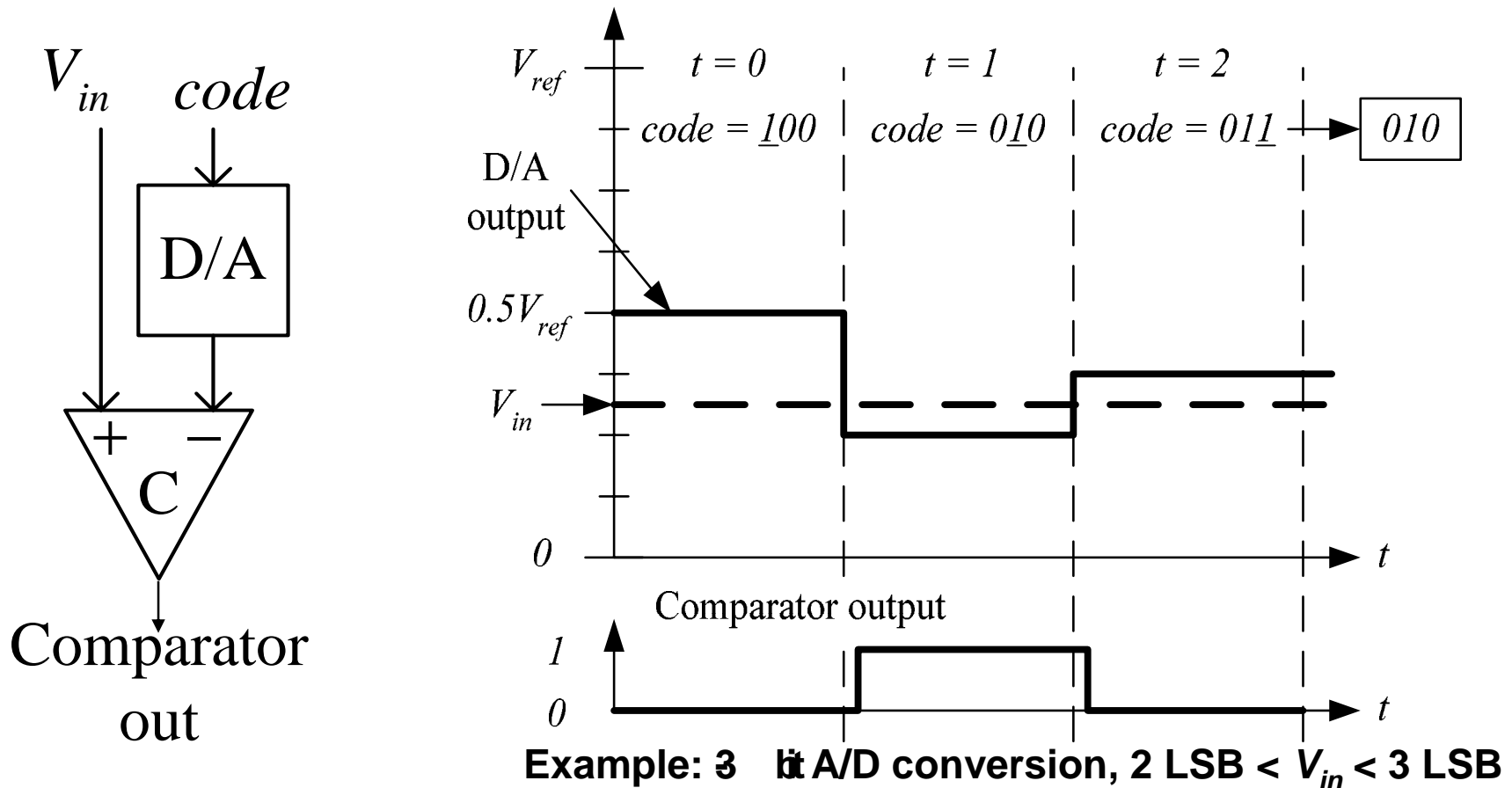




# Successive Approximation A/D

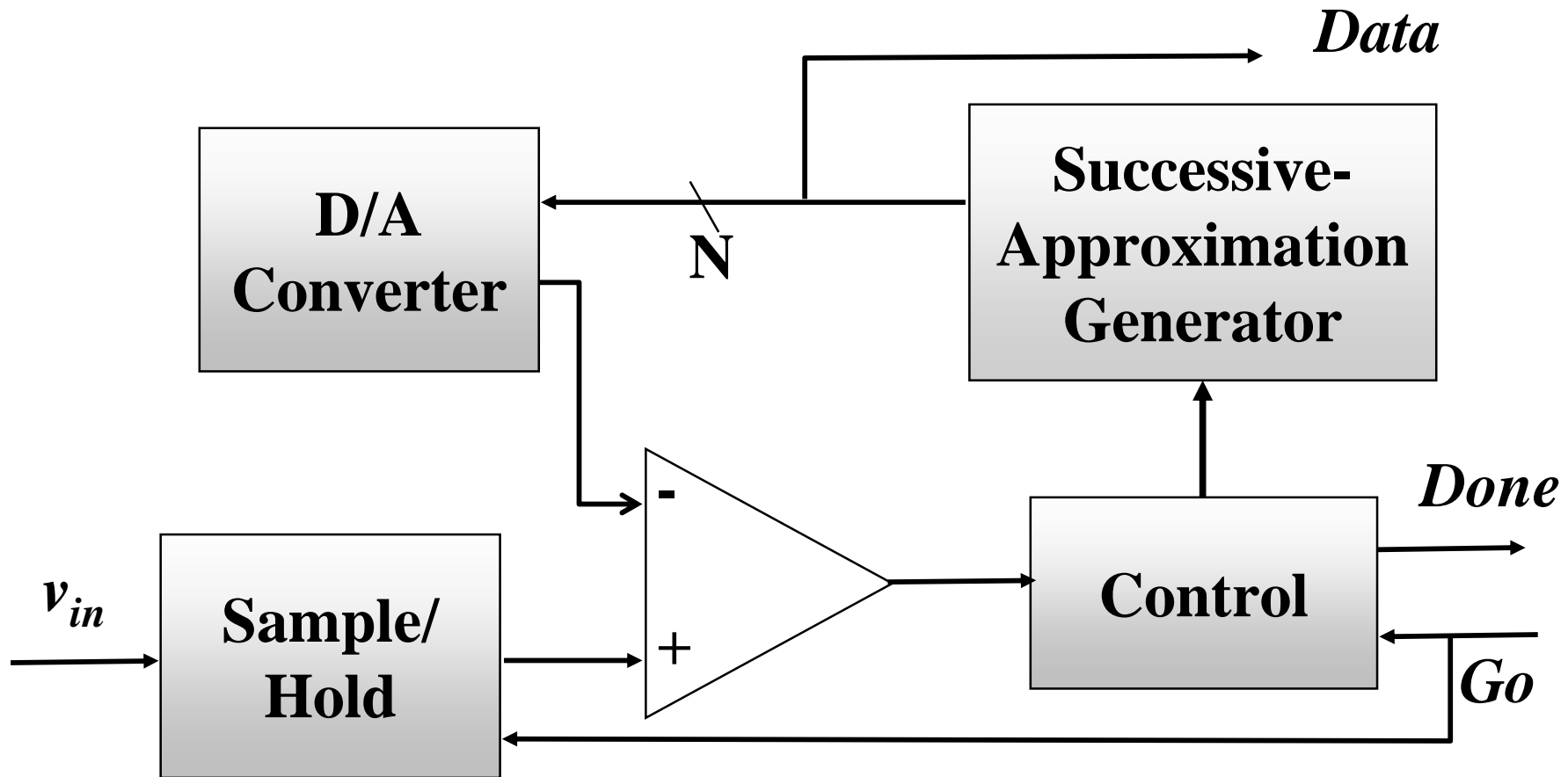


- D/A converters are typically compact and easier to design. A successive approximation A/D uses a D/A converter and a comparator.
- D to A generates analog voltage which is compared to the input voltage.
- If D to A voltage > input voltage, then set that bit; otherwise, reset that bit.
- This type of A to D takes a fixed amount of time proportional to the bit length.





# Successive Approximation A/D



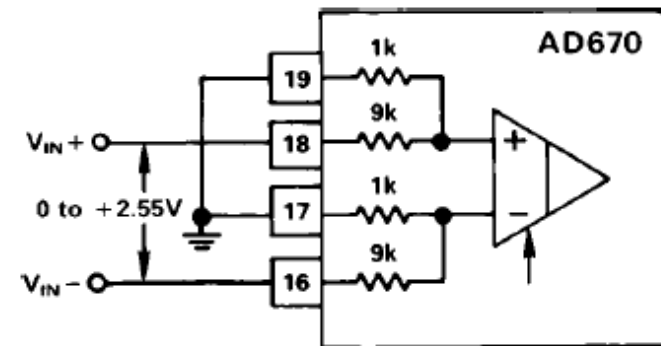
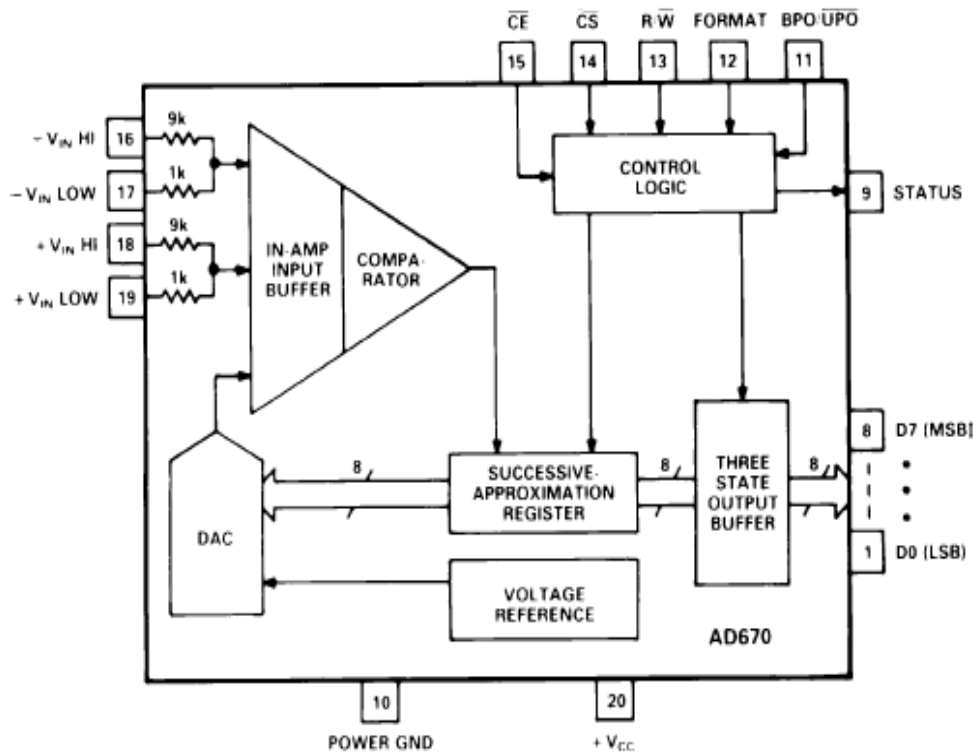
- Serial conversion takes a time equal to  $N(t_{D/A} + t_{comp})$ .



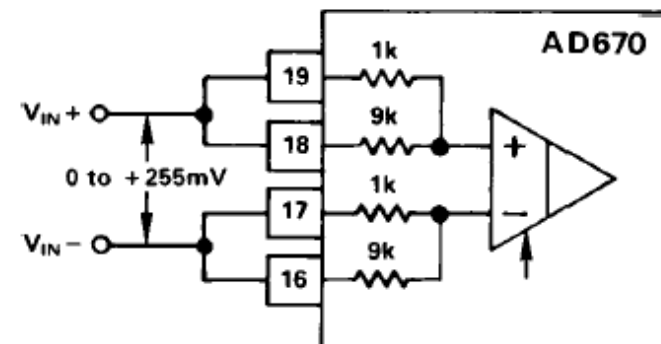
# Successive Approximation A/D (AD670) – Used in Lab 3



- We will use the Unipolar configuration: Pin 11 should be low.



2a. 0 V to 2.55 V (10 mV/LSB)



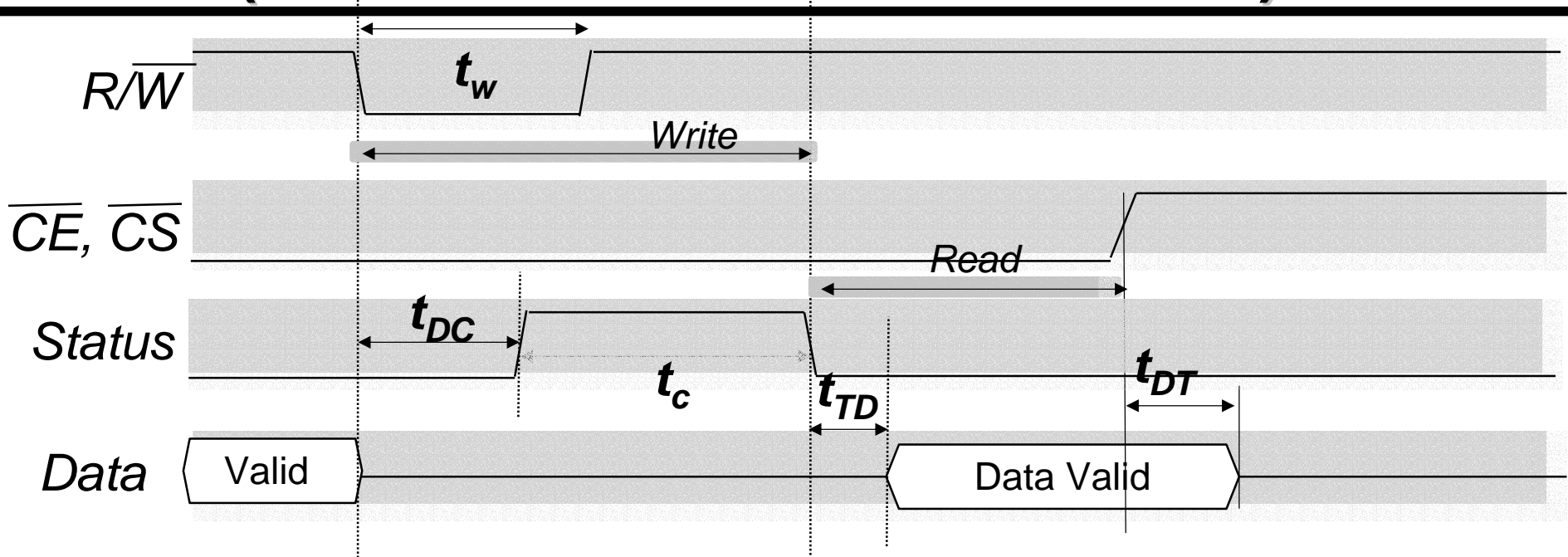
2b. 0 mV to 255 mV (1 mV/LSB)

BPO/ $\overline{UPD}$	FORMAT	INPUT RANGE/ OUTPUT FORMAT
0	0	Unipolar/Straight Binary
1	0	Bipolar/Offset Binary
0	1	Unipolar/2s Complement
1	1	Bipolar/2s Complement

- 10 $\mu$ s conversion time



# Single Write, Single Read Operation (see data sheet for other modes)



$t_w$  (write/start pulse width) = 300ns (min)

$t_{DC}$  (delay to start conversion) = 700ns (max)

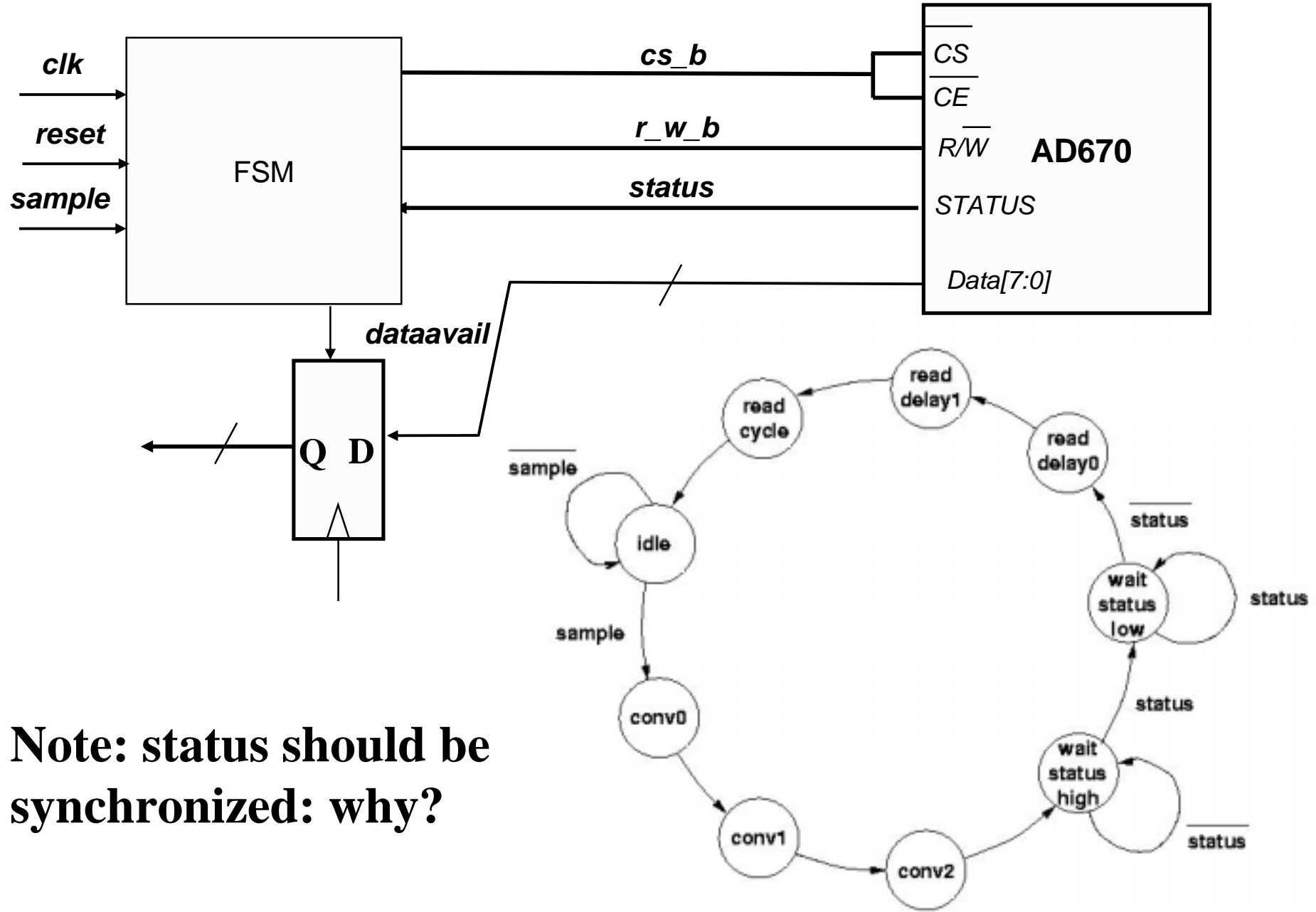
$t_c$  (conversion time) = 10 $\mu$ s (max)

$t_{TD}$  (Bus Access Time) = 250 (max)

$t_{TD}$  (Output Float Delay) = 150 (max)

- Control bits  $\overline{CE}$  and  $\overline{CS}$  can be wired to ground if A/D is the only chip driving the bus.
- Good idea to read the result of the last conversion and compute while waiting for the current conversion to finish (due to the long conversion time).

# Simple A/D Interface FSM



**Note: status should be synchronized: why?**



# Example A/D VHDL interface (courtesy J. Oey)



-- VHDL code for a/d interface

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity adinterface is port(

clk : in std\_logic;

-- User Interface

sample : in std\_logic;

dataavail : out std\_logic;

-- A/D Interface

r\_w\_b : out std\_logic;

cs\_b : out std\_logic;

status : in std\_logic;

-- reset

reset : in std\_logic);

end adinterface;

architecture adarch of adinterface is

signal statussync : std\_logic;

signal rwinternal : std\_logic;

signal csinternal : std\_logic;

signal present\_state : std\_logic\_vector(3 downto 0);

signal next\_state : std\_logic\_vector(3 downto 0);

constant idle : std\_logic\_vector(3 downto 0) := "0000";

constant conv0 : std\_logic\_vector(3 downto 0) := "0001";

constant conv1 : std\_logic\_vector(3 downto 0) := "0010";

constant conv2 : std\_logic\_vector(3 downto 0) := "0011";

constant waitstatushigh : std\_logic\_vector(3 downto 0) :=  
"0100";

constant waitstatuslow : std\_logic\_vector(3 downto 0) := "0101";

constant readdelay0 : std\_logic\_vector(3 downto 0) := "0110";

constant readdelay1 : std\_logic\_vector(3 downto 0) := "0111";

constant readcycle : std\_logic\_vector(3 downto 0) := "1000";

begin

clocked:process(clk)

begin

if rising\_edge(clk) then

if (reset = '1') then

present\_state <= idle;

else

present\_state <= next\_state;

end if;

-- synchronizing inputs and outputs

statussync <= status;

r\_w\_b <= rwinternal;

cs\_b <= csinternal;

end if;

end process;



# Example VHDL of A/D interface (II)



```
statecomb:process(present_state, sample, statussync)
```

```
begin
```

```
case present_state is
```

```
when idle =>
```

```
    rwinternal <= '1';
```

```
    csinternal <= '1';
```

```
    dataavail <= '0';
```

```
    if sample = '1' then
```

```
        next_state <= conv0;
```

```
    else
```

```
        next_state <= idle;
```

```
    end if;
```

```
when conv0 =>
```

```
    rwinternal <= '0';
```

```
    csinternal <= '0';
```

```
    dataavail <= '0';
```

```
    next_state <= conv1;
```

```
when conv1 =>
```

```
    rwinternal <= '0';
```

```
    csinternal <= '0';
```

```
    dataavail <= '0';
```

```
    next_state <= conv2;
```

```
when conv2 =>
```

```
    rwinternal <= '0';
```

```
    csinternal <= '0';
```

```
    dataavail <= '0';
```

```
    next_state <= waitstatushigh;
```

```
when waitstatushigh =>
```

```
    rwinternal <= '0';
```

```
    csinternal <= '0';
```

```
    dataavail <= '0';
```

```
    if statussync = '1' then
```

```
        next_state <= waitstatuslow;
```

```
    else
```

```
        next_state <= waitstatushigh;
```

```
    end if;
```

```
when waitstatuslow =>
```

```
    rwinternal <= '1';
```

```
    csinternal <= '0';
```

```
    dataavail <= '0';
```

```
    if statussync = '0' then
```

```
        next_state <= readdelay0;
```

```
    else
```

```
        next_state <= waitstatuslow;
```

```
    end if;
```

```
when readdelay0 =>
```

```
    rwinternal <= '1';
```

```
    csinternal <= '0';
```

```
    dataavail <= '0';
```

```
    next_state <= readdelay1;
```

```
when readdelay1 =>
```

```
    rwinternal <= '1';
```

```
    csinternal <= '0';
```

```
    dataavail <= '0';
```

```
    next_state <= readcycle;
```

```
when readcycle =>
```

```
    rwinternal <= '1';
```

```
    csinternal <= '0';
```

```
    dataavail <= '1';
```

```
    next_state <= idle;
```

```
when others =>
```

```
    rwinternal <= '1';
```

```
    csinternal <= '1';
```

```
    dataavail <= '0';
```

```
    next_state <= idle;
```

```
end case;
```

```
end process;
```

```
end adarch;
```



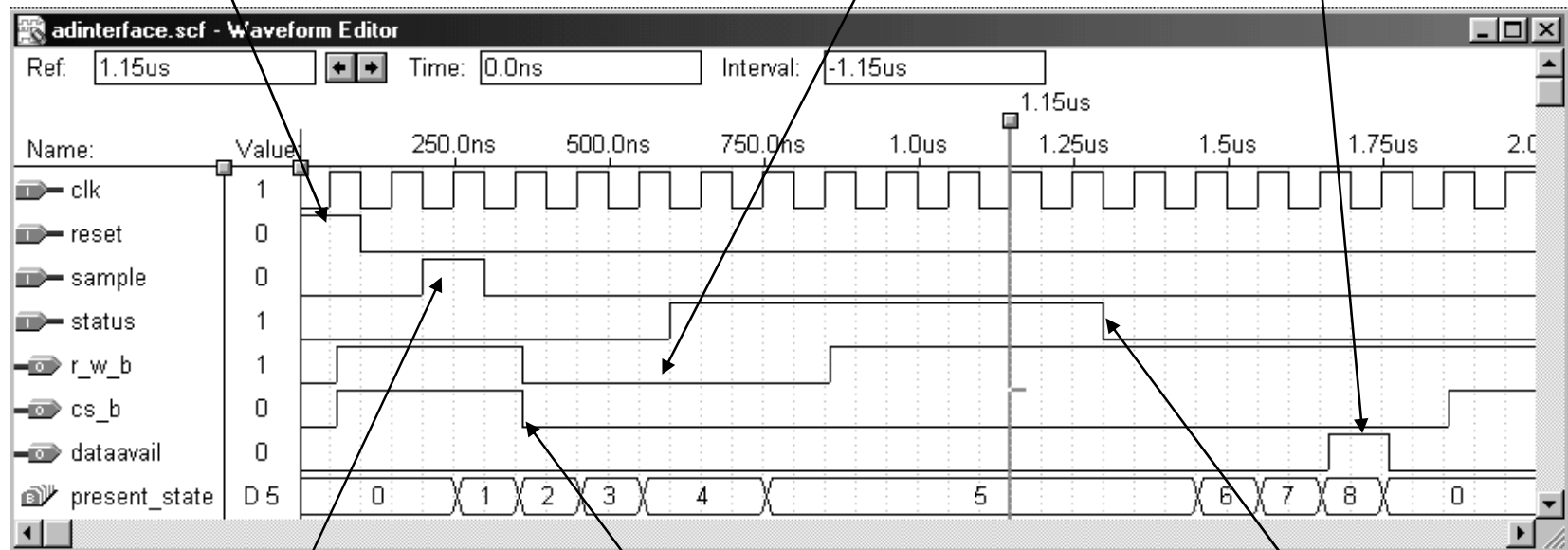
# Simulation



**r\_w\_b must stay low for at least 3 cycles (@ 100ns period) –  
Can bring back high in the waitstatushigh state**

**On reset, present state goes to 0.**

**Enable read flip-flop**



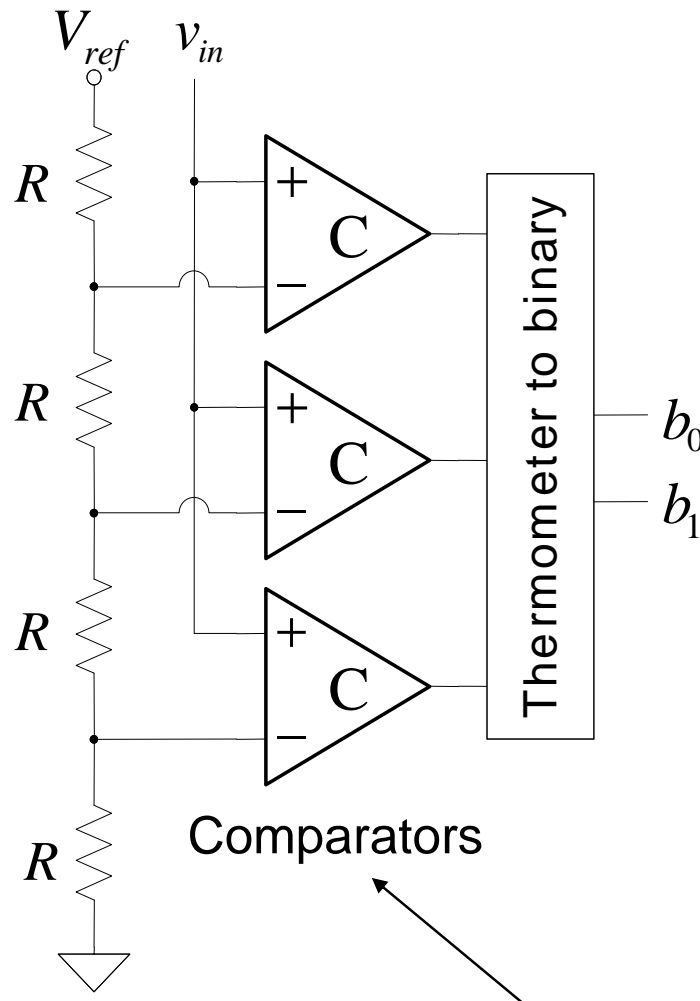
**Sample pulse initiates data conversion.**

**Wait for ~10 $\mu$ s for status to go low.**

**Notice a one-cycle delay since A/D control signal is clocked.**



# Flash A/D Converter

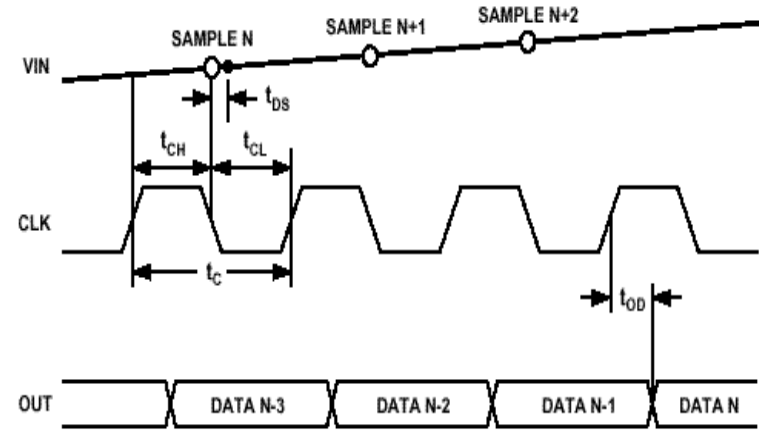
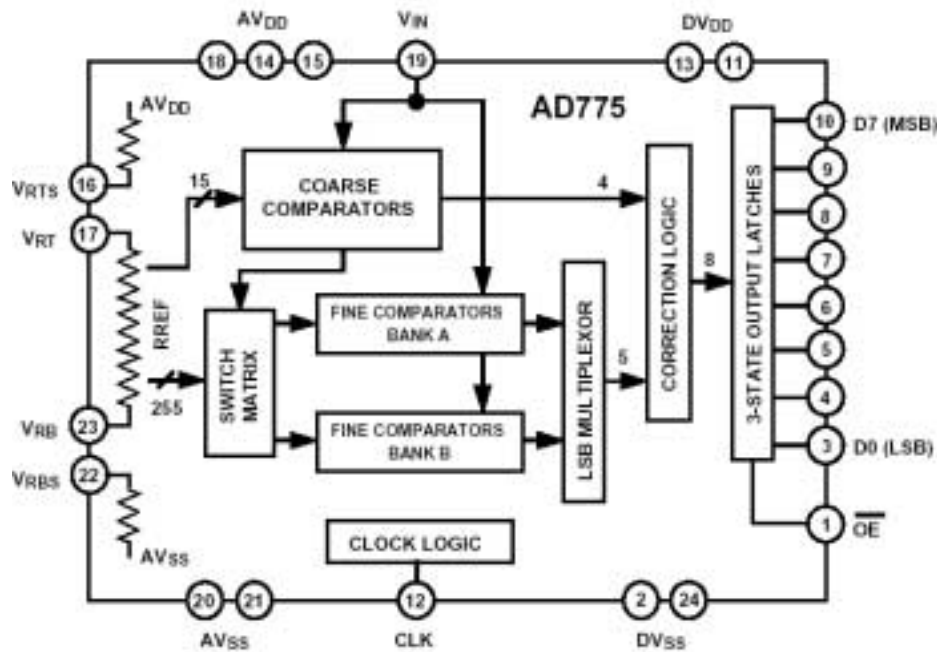


- Brute force A/D conversion
- Simultaneously compare the analog value with every possible reference value.
- Fastest method of A/D conversion
- Size scales exponentially with precision (requires  $2^N$  comparators).

**Another example of OpAmp in open loop**



# AD 775 – Flash Data Converter



## TIMING SPECIFICATIONS

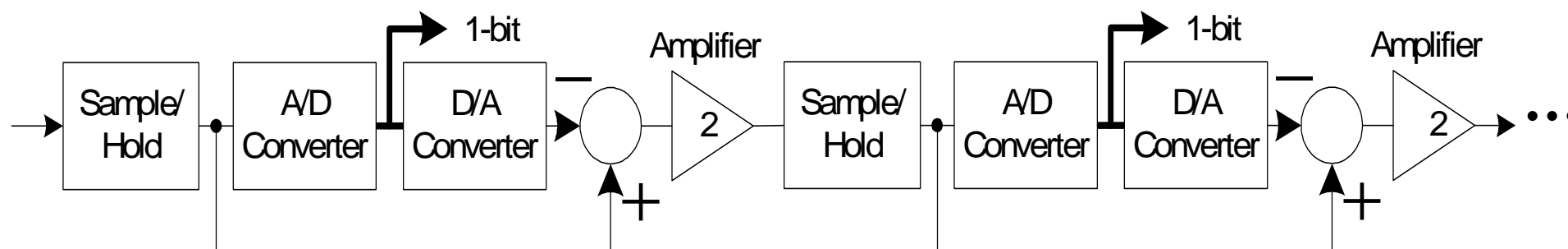
	Symbol	Min	Typ	Max	Units
Maximum Conversion Rate		20	35		MHz
Clock Period	$t_c$	50			ns
Clock High	$t_{CH}$	25			ns
Clock Low	$t_{CL}$	25			ns
Output Delay	$t_{OD}$		18	30	ns
Pipeline Delay (Latency)				2.5	Clock Cycles
Sampling Delay	$t_{DS}$		4		ns
Aperture Jitter			30		ps



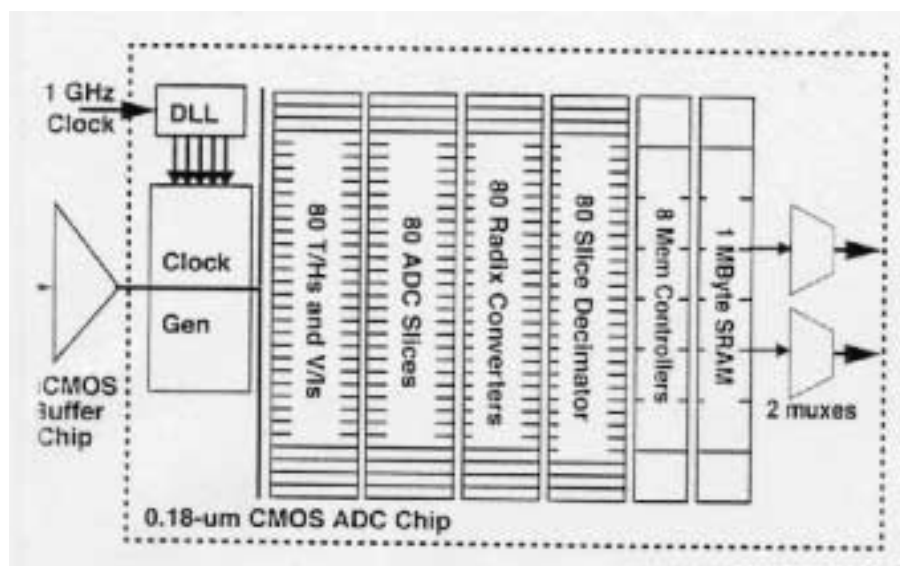
# High Performance Converters: Use Pipelining and Parallelism!



**Pipelining (used in video rate, RF basestations, etc.)**



**Parallelism (use many slower A/Ds in parallel to build very high speed A/D converters)**



**[ISSCC 2003],  
Poulton et. al.**

**20Gsample/sec,  
8-bit ADC  
from Agilent Labs**