



L12: Analog Building Blocks (OpAmps, A/D, D/A)



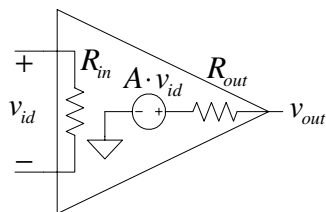
Acknowledgement: Dave Wentzloff



Introduction to Operational Amplifiers



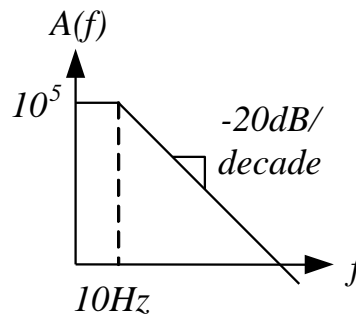
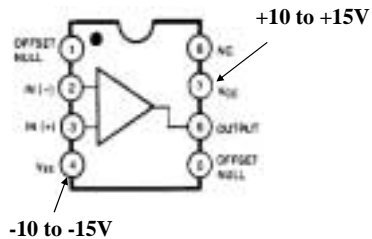
DC Model

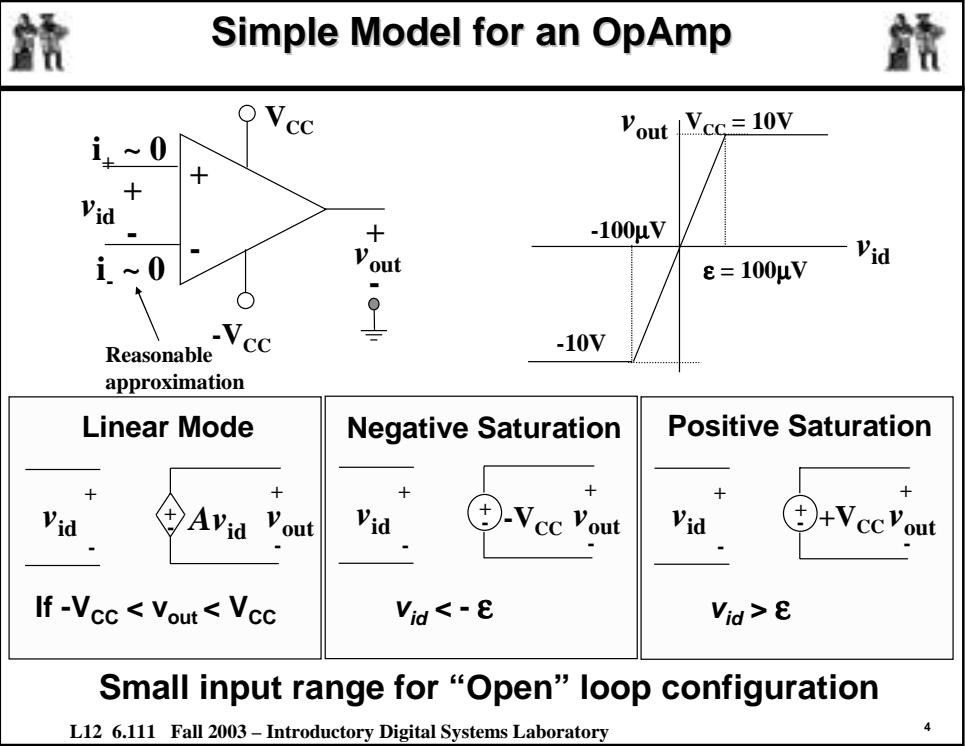
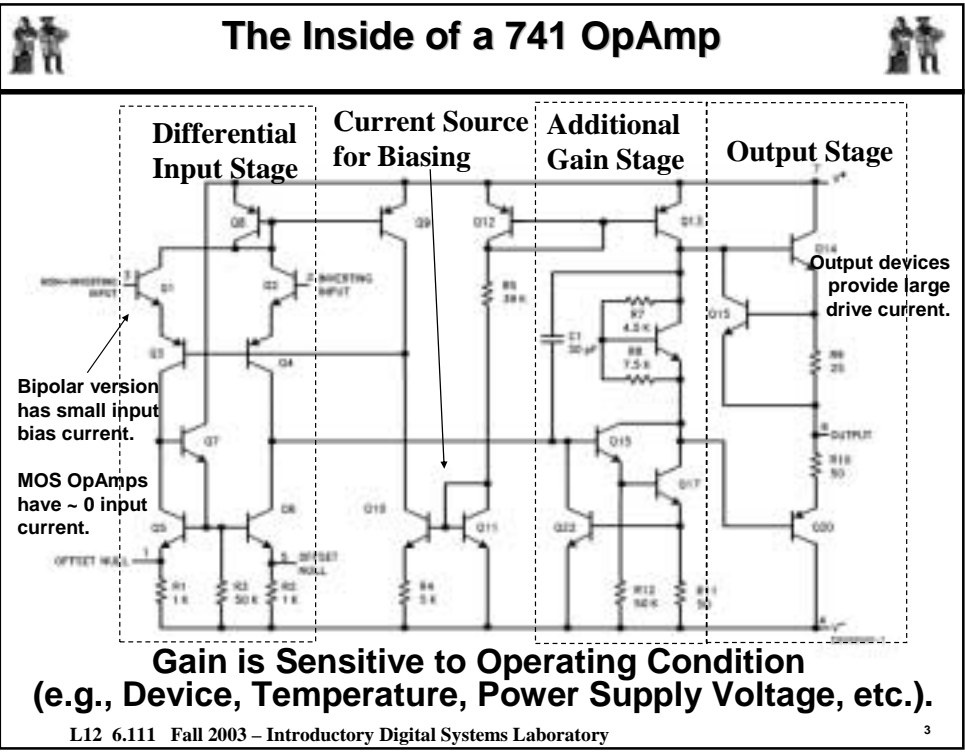


- Typically very high input resistance $\sim 300\text{K}\Omega$
- High DC gain ($\sim 10^5$)
- Output resistance $\sim 75\Omega$

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

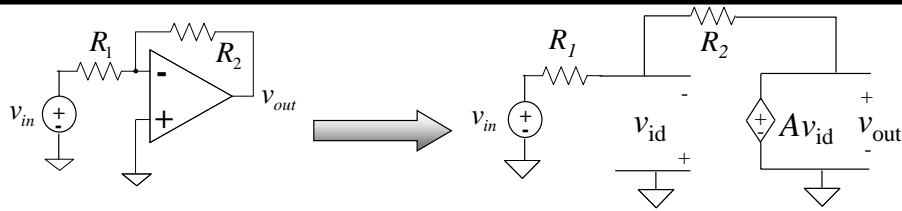
LM741 Pinout







The Power of Feedback



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

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

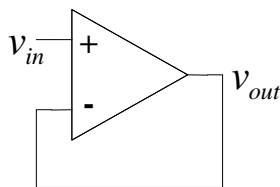
- Overall (closed loop) gain does not depend on open loop gain.
- Trade gain for robustness.
- Easier analysis approach: “virtual short circuit approach”
 - $v_+ = v_- = 0$ if OpAmp is linear



Basic OpAmp Circuits

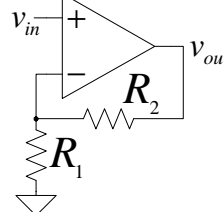


Follower



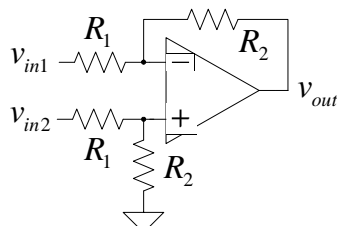
$$v_{out} \approx v_{in}$$

Non-inverting



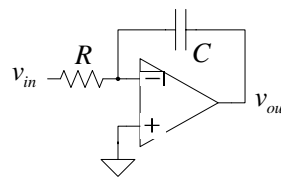
$$v_{out} \approx \frac{R_1 + R_2}{R_1} v_{in}$$

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

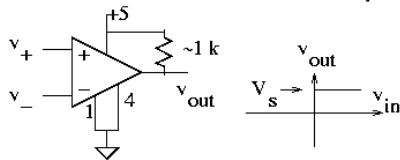


Analog Comparator:

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

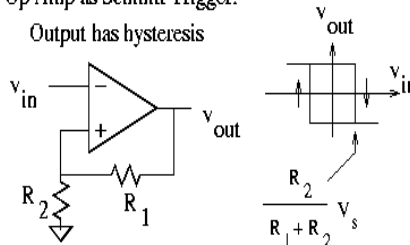


LM311 is a single supply comparator.

Schmitt Trigger:

Squares up signals

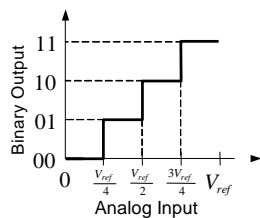
Op Amp as Schmitt Trigger:
Output has hysteresis



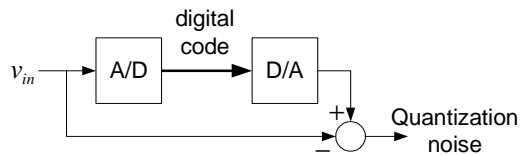
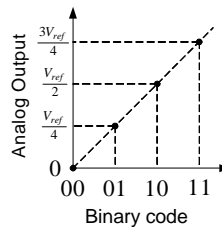
Data Conversion: Quantization Noise



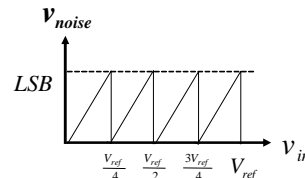
A/D Conversion



D/A Conversion



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

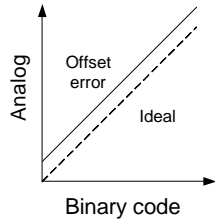




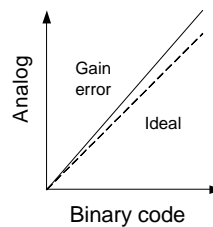
Non-idealities in Data Conversion



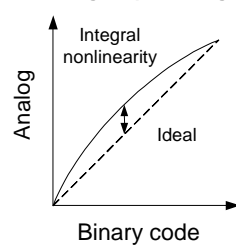
Offset – a constant voltage offset that appears at the output when the digital input is 0



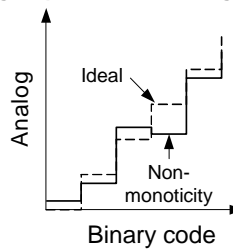
Gain error – deviation of slope from ideal value of 1



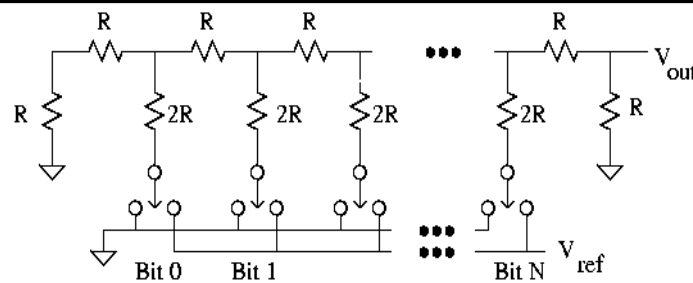
Integral nonlinearity – maximum deviation from the ideal analog output voltage



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-2R resistor ladder
- Settling time 1 μ s
- 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	10.00 V Range
0000 0000	00	0	0	0
0000 0001	01	1	0.016 V	0.078 V
0000 0010	02	2	0.032 V	0.156 V
0000 0011	03	3	0.048 V	0.234 V
0001 0000	10	16	0.160 V	0.825 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

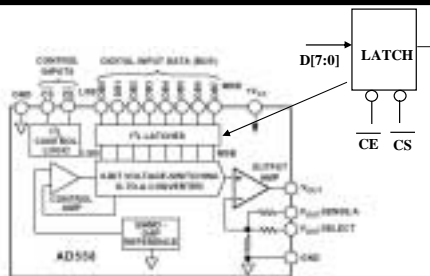
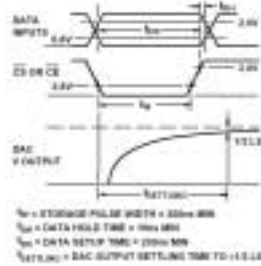


Table I. AD558 Control Logic Truth Table

Input Data	CE	CS	DAC Data	Latch Condition
0	0	0	0	"Transparent"
1	0	0	1	"Transparent"
0	0	0	0	Latching
1	0	0	1	Latching
0	0	0	0	Latching
1	0	0	1	Latching
X	1	X	Previous Data	Latched
X	X	1	Previous Data	Latched

NOTES
 X = Don't care state
 0 = Logic Threshold at Positive-Going Transition



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

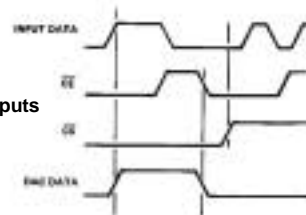
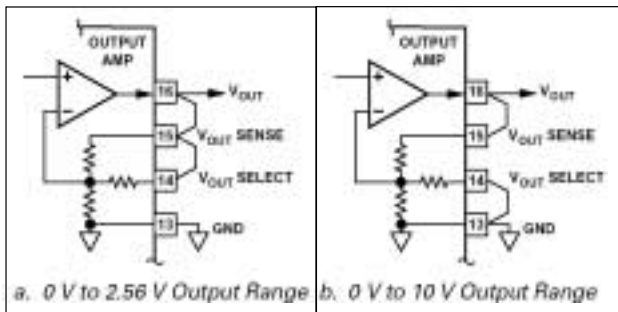


Figure 8. AD558 Control Logic Function



Setting the Voltage Range



Very similar to a non-inverting amp

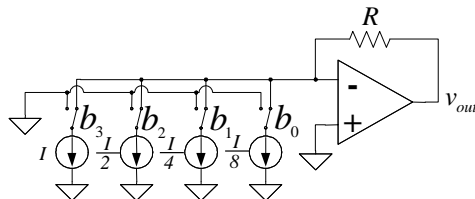
Strap output for different voltage ranges

Input Logic Coding

Digital Input Code			Output Voltage	
Binary	Hexadecimal	Decimal	2.56 V Range	10.00 V Range
0000 0000	00	0	0	0
0000 0001	01	1	0.016 V	0.030 V
0000 0010	02	2	0.032 V	0.078 V
0000 0011	03	3	0.048 V	0.102 V
0000 0100	04	4	0.064 V	0.150 V
0000 0101	05	5	0.080 V	0.174 V
0000 0110	06	6	0.096 V	0.222 V
0000 0111	07	7	0.112 V	0.246 V
0001 0000	08	8	0.128 V	0.294 V
0001 0001	09	9	0.144 V	0.318 V
0001 0010	0A	10	0.160 V	0.366 V
0001 0011	0B	11	0.176 V	0.390 V
0001 0100	0C	12	0.192 V	0.438 V
0001 0101	0D	13	0.208 V	0.462 V
0001 0110	0E	14	0.224 V	0.510 V
0001 0111	0F	15	0.240 V	0.534 V
0010 0000	10	16	0.256 V	0.582 V
0010 0001	11	17	0.272 V	0.606 V
0010 0010	12	18	0.288 V	0.654 V
0010 0011	13	19	0.304 V	0.678 V
0010 0100	14	20	0.320 V	0.726 V
0010 0101	15	21	0.336 V	0.750 V
0010 0110	16	22	0.352 V	0.798 V
0010 0111	17	23	0.368 V	0.822 V
0011 0000	18	24	0.384 V	0.870 V
0011 0001	19	25	0.400 V	0.894 V
0011 0010	1A	26	0.416 V	0.942 V
0011 0011	1B	27	0.432 V	0.966 V



Another Approach: Binary-Weighted DAC

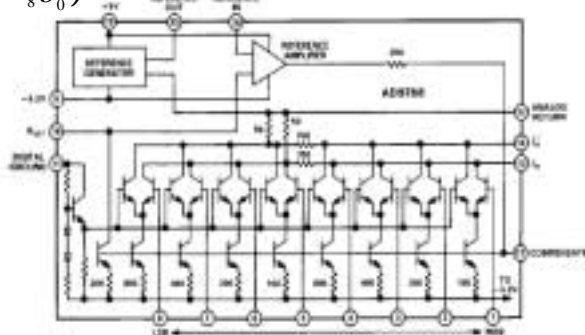


- Switch binary-weighted currents
- MSB to LSB current ratio is 2^N .

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

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

AD9768



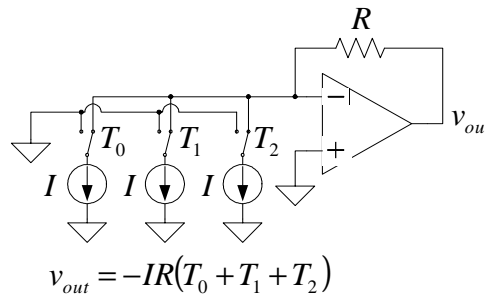
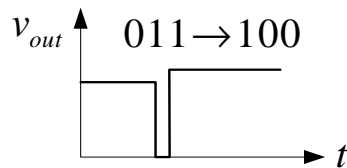


Glitching and Thermometer D/A



- 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.

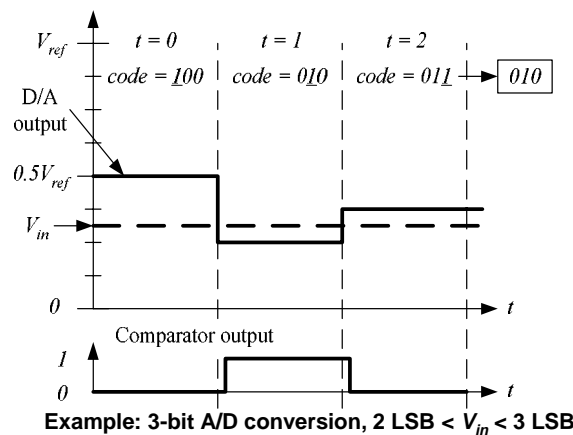
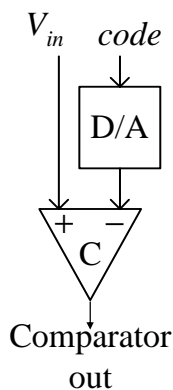
Binary		Thermometer		
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1



Successive-Approximation A/D

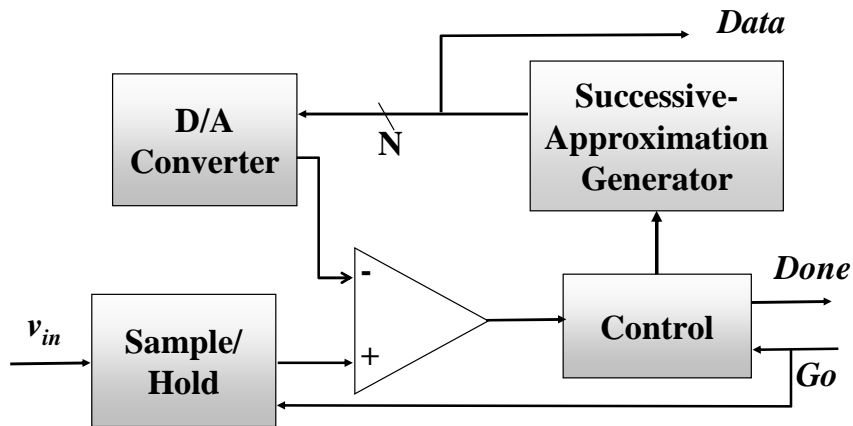


- D/A converters are typically compact and easier to design. A successive-approximation A/D uses a D/A converter and a comparator.
- D to A generates analog voltage which is compared to the input voltage.
- If D to A voltage > input voltage, then set that bit; otherwise, reset that bit.
- This type of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





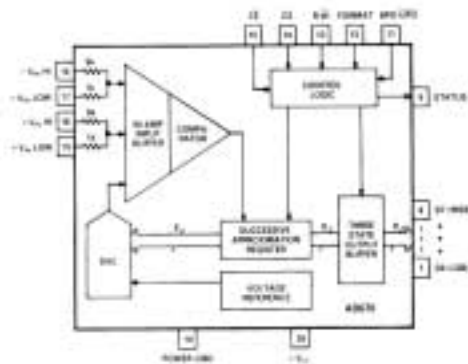
Successive-Approximation A/D



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



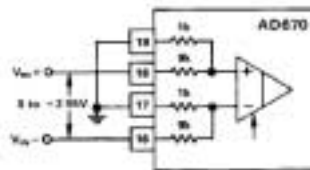
Successive-Approximation A/D (AD670) – Used in Lab 3



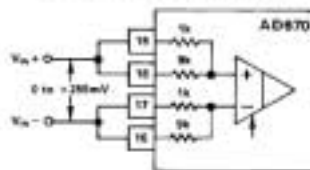
BPO(UN)	FORMAT	INPUT RANGE/ OUTPUT FORMAT
0	0	Unipolar/Straight Binary
1	0	Bipolar/Offset Binary
0	1	Unipolar/2s Complement
1	1	Bipolar/2s Complement

- 10 μ s conversion time

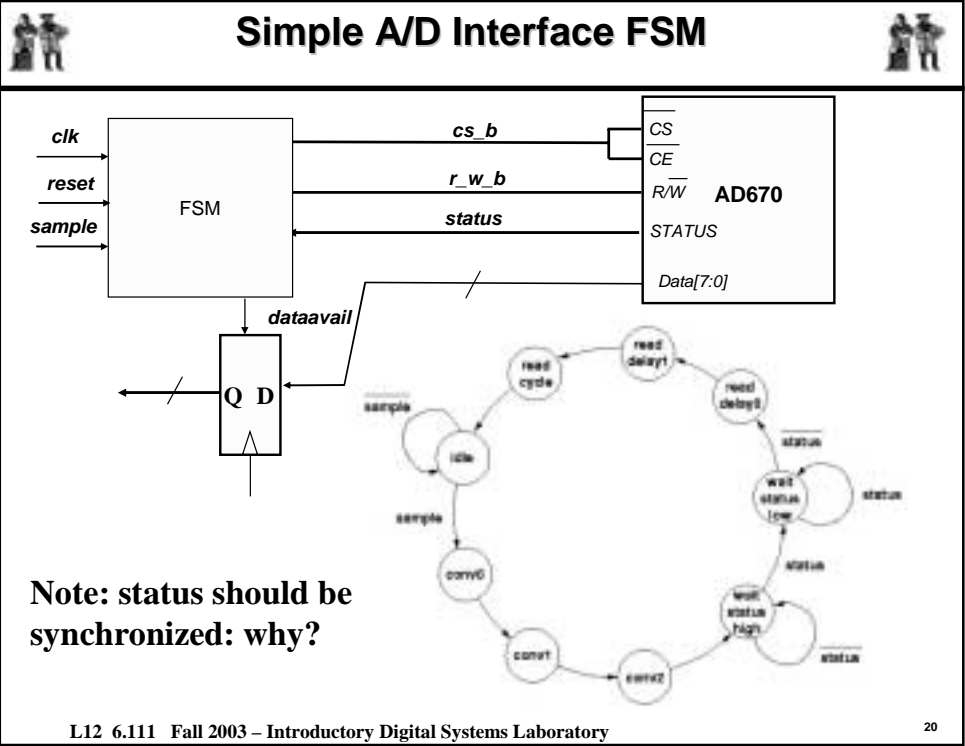
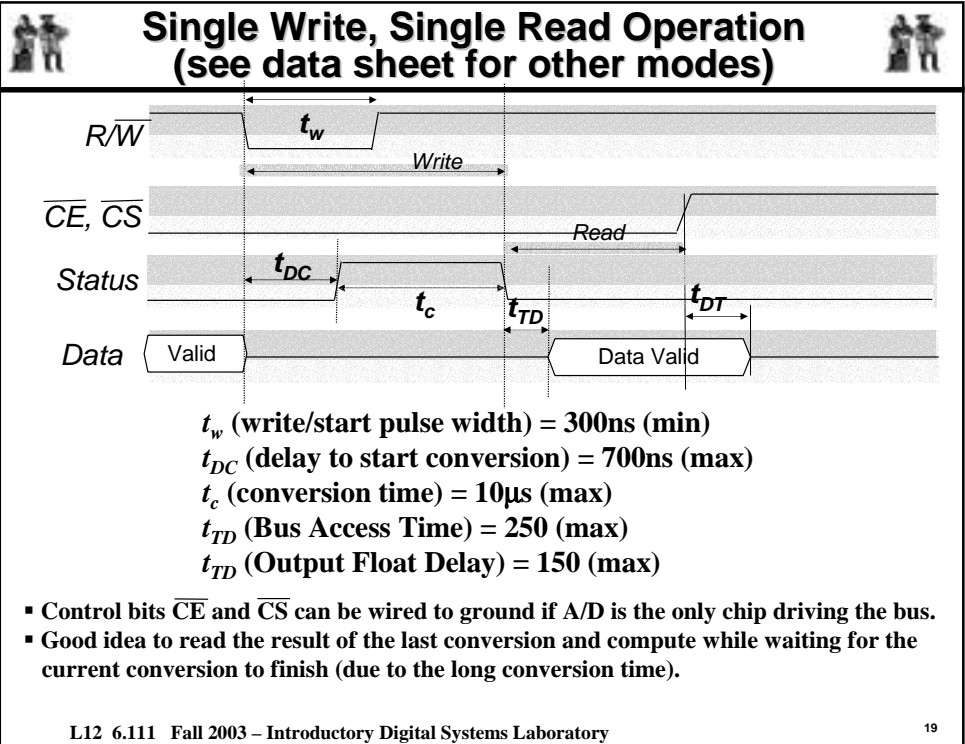
- We will use the Unipolar configuration: Pin 11 should be low.



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



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





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;
end adarch;

```



Example VHDL of A/D interface (II)



```

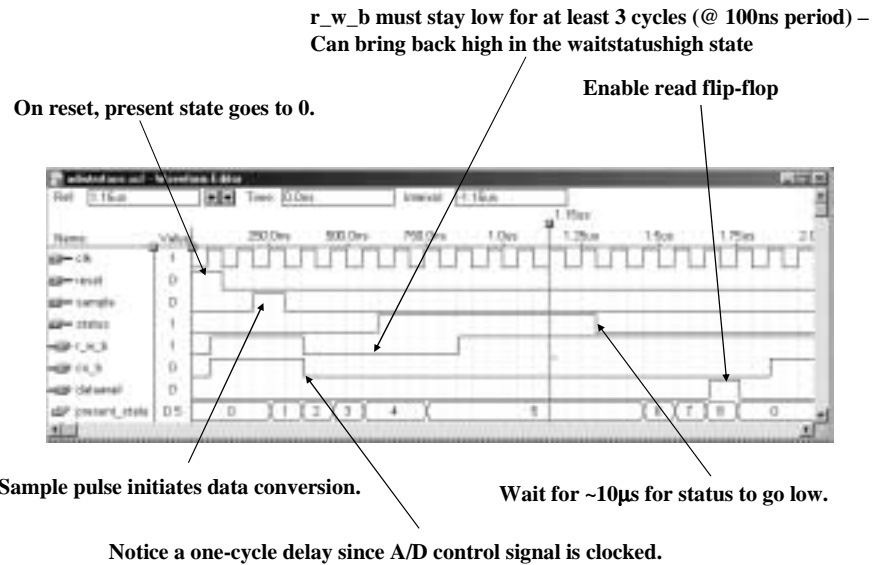
statecomb:process(present_state, sample, statussync)
begin
    case present_state is
        when idle =>
            rwinternal <= '1';
            csinternal <= '1';
            dataavail <= '0';
            if sample = '1' then
                next_state <= conv0;
            else
                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;

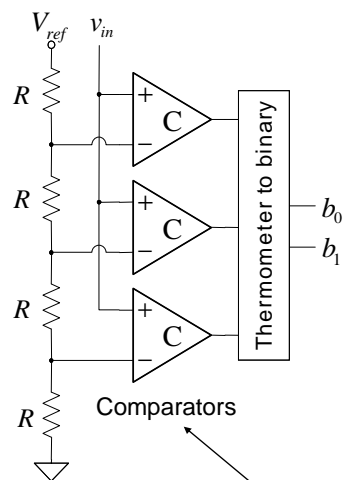
```



Simulation



Flash A/D Converter

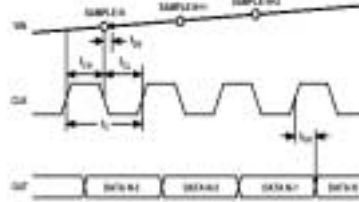
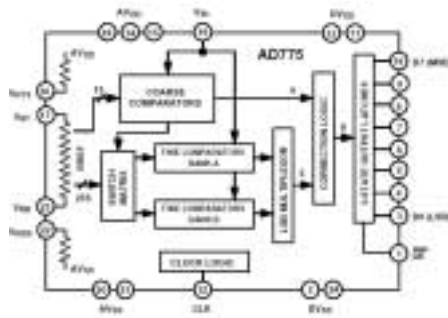


- Brute-force A/D conversion
- Simultaneously compare the analog value with every possible reference value.
- Fastest method of A/D conversion
- Size scales exponentially with precision (requires 2^N comparators).

Another example of OpAmp in open loop



AD 775 – Flash Data Converter



TIMING SPECIFICATIONS

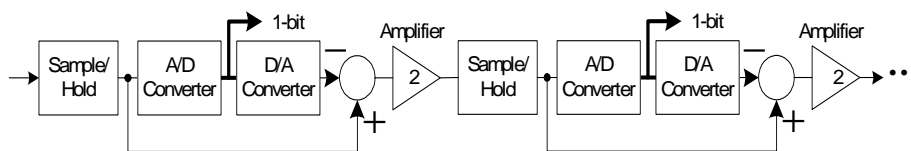
	Symbol	Min	Typ	Max	Units
Minimum Conversion Rate		20	25		MHz
Clock Period	t_c	50			ns
Clock High	t_{cH}	25			ns
Clock Low	t_{cL}	25			ns
Output Delay	t_{out}		18	30	ns
Pipeline Delay (Latency)				2.5	Clock Cycles
Sampling Delay	t_{in}		4		ns
Aperture Jitter			30		ps



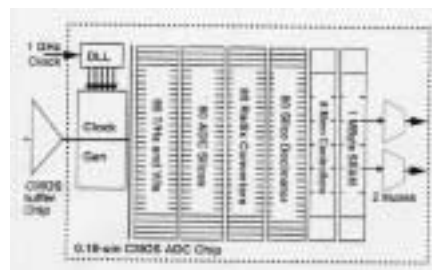
High Performance Converters: Use Pipelining and Parallelism!



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