ANALOG INPUT SYSTEMS

EE3232 DIGITAL SYSTEMS III
CLASS NOTES CHAPTER 8

Department of Electrical Engineering
University of New Brunswick

 January 5, 2000

SUMMARY

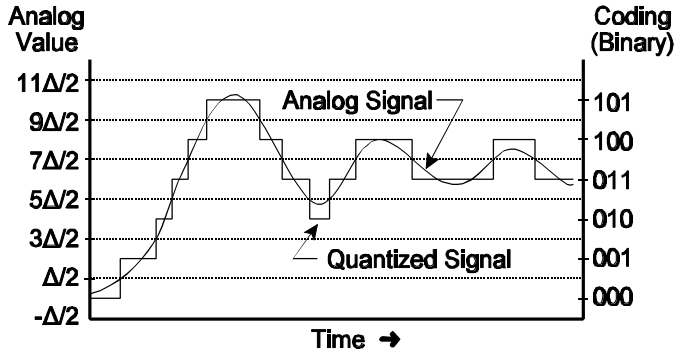
- Objectives.
- Quantization and coding.
- Examples of analog sensors.
- Ideal ADC transfer function.
- Comparators.
- A 1-bit ADC.
- Flash ADC's.
- Counting ADC's.
- Tracking ADC's.
- Successive approximation ADC's.
- ADC specifications.
- ADC calibration.
- Interfacing ADC's to a system bus.
- Analog multiplexers.
- Multi-channel analog input systems.

OBJECTIVES

- To explain the analog conversion errors caused by quantization.
- To draw the logic diagram of the internal architecture and explain the operation of the analog to digital converters,
  - Flash ADC,
  - Counting ADC,
  - Tracking ADC and
  - Successive approximation ADC.
- To develop the H/W and S/W for interfacing an ADC to a system bus.
- To develop the hardware and software for interfacing a multi-channel analog input system to a system bus.

QUANTIZATION AND CODING

- Converting an analog value into a digital representation consists of two operations: ***quantization*** and ***coding***.
- Quantization divides the range of the analog value into a fixed number of quantization intervals.
- Within each interval the analog value is mapped to one value or quanta = the midpoint of the quantization interval.
- Each quanta is then assigned a unique coding such as straight binary, offset binary, two's complement, etc.

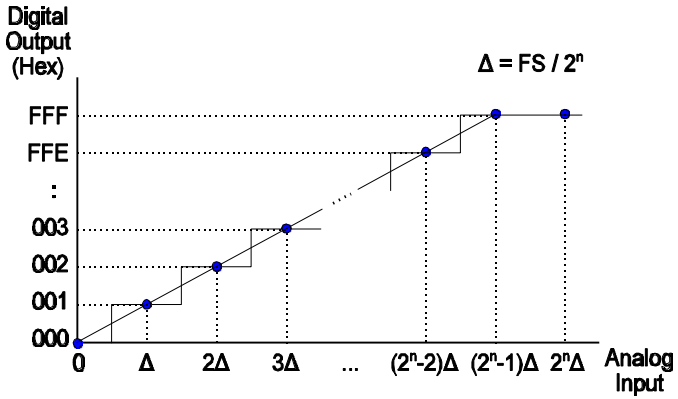


EXAMPLES OF ANALOG SENSORS

- Light sensors, Infrared sensors
- Temperature sensors,
- Strain gauges,
- Tachometers, Accelerometers,
- Potentiometers,
- Pressure, Level, Flow sensors,
- pH sensors, Conductivity sensors,
- Load and Force sensors,
- Displacement sensors.

IDEAL ADC TRANSFER FUNCTIoN (STRAIGHT BINARY)

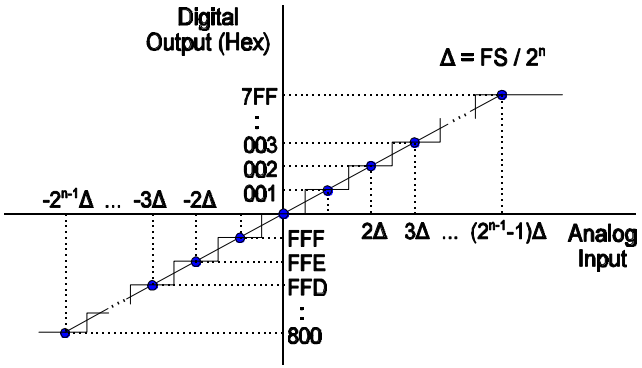
- A unipolar, straight binary ADC transfer function.



- If the analog input is in the range { mΔ ± Δ/2 } then the ADC output code = m, m ∈ unsigned = { 0, 2^n - 1 }.

IDEAL ADC TRANSFER FUNCTION (TWO'S COMPLEMENT)

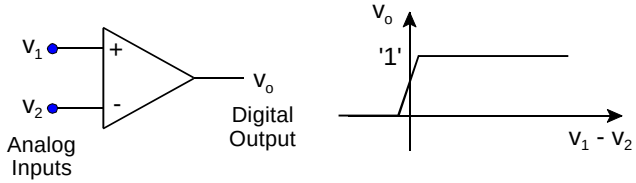
- A bipolar, two's complement ADC transfer function.



- If the analog input is in the range { mΔ ± Δ/2 } then the ADC output code = m, m ∈ signed = { -2^{n-1}, 2^{n-1} - 1 }.

COMPARATORS

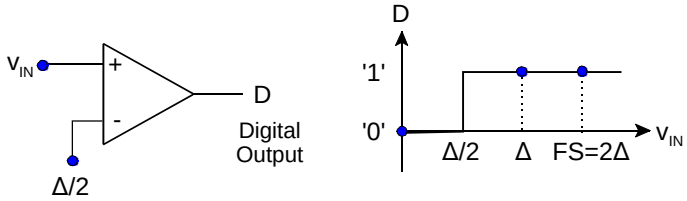
- **Operation of a Comparator**
  - v_o = '1' , if $v_1 > v_2$
  - v_o = '0' , if $v_2 > v_1$



- Construct a 1-bit ADC using a comparator.

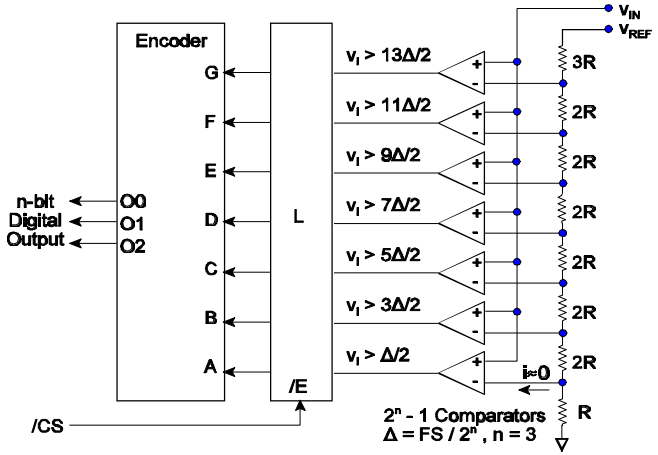
A 1-BIT ADC USING A COMPARATOR

- n = 1, Δ = FS ÷ 2^n



FLASH ADC'S

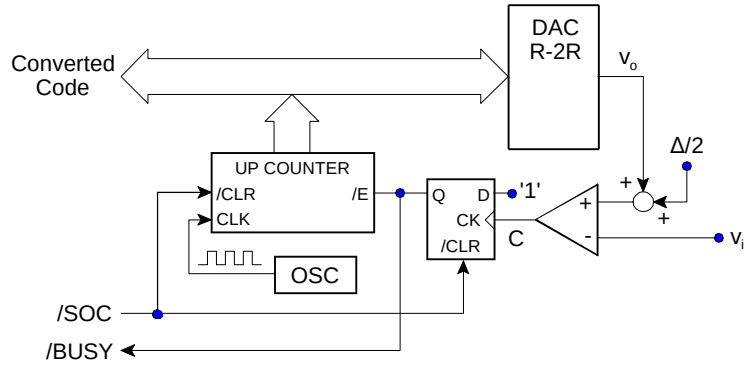
- Flash ADC's have very fast conversion times - 10's of ns.



- To achieve maximum possible conversion rates - very high speed microprocessors or dedicated hardware is required.
- A flash ADC may be interfaced to a microcomputer system bus via a simple parallel input port.
- Give the truth table for the encoder of a straight binary flash ADC.
- How would you alter the hardware to implement a bipolar flash ADC ?

COUNTING ADC'S

- Put a DAC in a feedback loop.
- Adjust the input of DAC until there is a match between the DAC output and the analog signal to be converted.



- **Operation of a Counting ADC :**

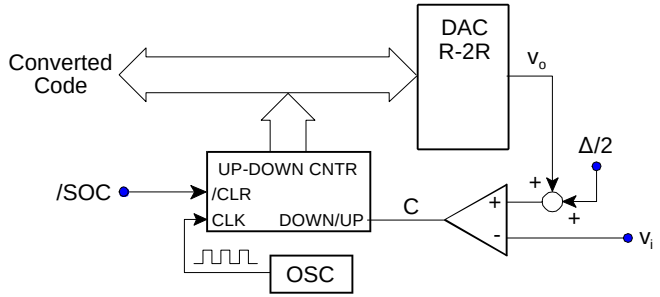
1. /SOC, start of conversion is activated,
   - Clearing the counter and
   - Resetting (activating) the /BUSY flip flop.

2. The counter is now enabled,
   - Counter increments with each clock until $v_o + \Delta/2$ just exceeds the analog input, v_i.
   - The comparator OP, C, switches from '0' to '1' setting (deactivating) the /BUSY flip flop.
   - The Converted Code is available at the output of the counter.

- **NOTE :** v_i must be constant during analog conversion otherwise conversion errors may occur.

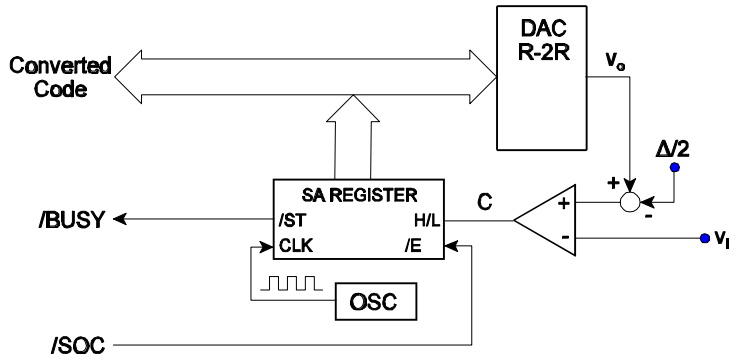
TRACKING ADC'S

- **Operation of a Tracking ADC :**
  - A variation of the counting ADC :
  - An up/down counter replaces the binary up counter.
  - If C = '1', then the counter counts down.
  - If C = '0', then the counter counts up.



SUCCESSIVE APPROXIMATION (SA) ADC'S

- A more intelligent algorithm for iterating the DAC input may reduce the conversion time!



SUCCESSIVE APPROXIMATION (SA) ADC'S (CONT'D)

- The conversion time for a counting ADC depends on the input, v_i.
- The conversion time for a successive approximation ADC is (for all practical purposes) independent of v_i.

- ***SA Register Operation for an N-bit Straight Binary ADC*** :
  - Reset SA REGISTER output to zero;
  - j = N - 1 ;
  - while (j >= 0) {
  - Set bit-j of SA REGISTER OP ;
  - If (H/L == '1') { Reset bit-j ; }
  - j = j - 1 ;
  - }

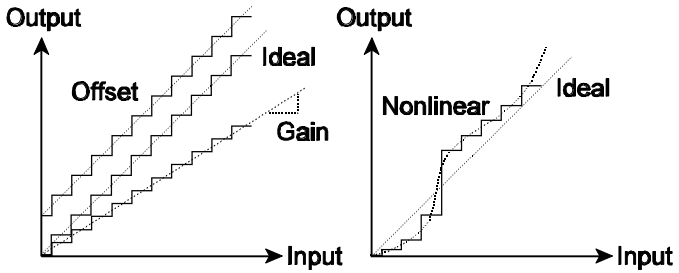
EXAMPLE

- ***Example*** : For a 16V, 4-bit, straight binary, successive approximation ADC, what is the sequence at the OP of the SA Register during the conversion of 7.9V? 7.2V?

Iteration	SA Reg OP	v_o - Δ/2	C
Vin = 7.9			
3.	1000 ➜ 1000	7.5 V	'0'
2.	1100 ➜ 1000	11.5 V	'1'
1.	1010 ➜ 1000	9.5 V	'1'
0.	1001 ➜ 1000	8.5 V	'1'
Vin = 7.2			
3.	1000 ➜ 0000	7.5 V	'1'
2.	0100 ➜ 0100	3.5 V	'0'
1.	0110 ➜ 0110	5.5 V	'0'
0.	0111 ➜ 0111	6.5 V	'0'

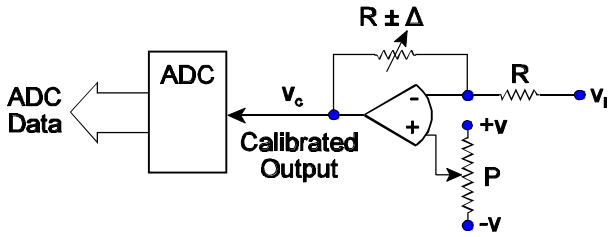
ADC SPECIFICATIONS

- Real ADC's deviate from the ideal due to,
  - Offset Errors,
  - Gain or Scale Factor Errors and
  - Nonlinear Errors.

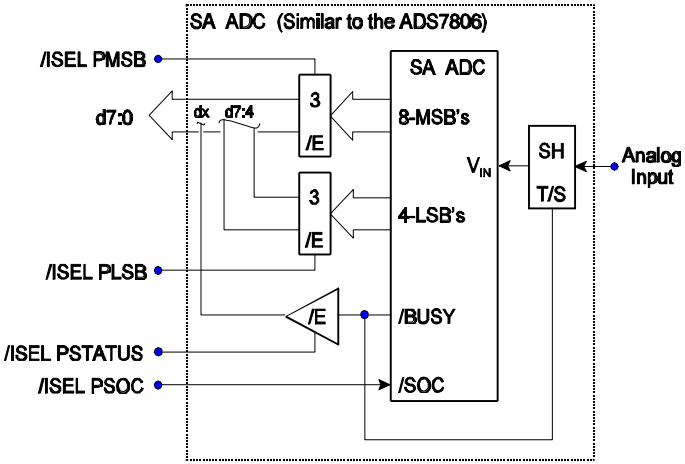


ADC CALIBRATION (NULLS GAIN AND OFFSET ERRORS)

- **Calibration Procedure** (For a straight binary ADC)
  - Set v_i = 0V ;
  - While (ADC Data != Code for Zero, e.g. 000..0B) {
  - Adjust P ;
  - }
  - Set v_i = Maximum Voltage, e.g. (2^n - 1) FS / 2^n ;
  - While (ADC data != Code for full scale, e.g. 111..1B) {
  - Adjust Δ ;
  - }



H/W INTERFACE BETWEEN A SA ADC AND A SYSTEM BUS

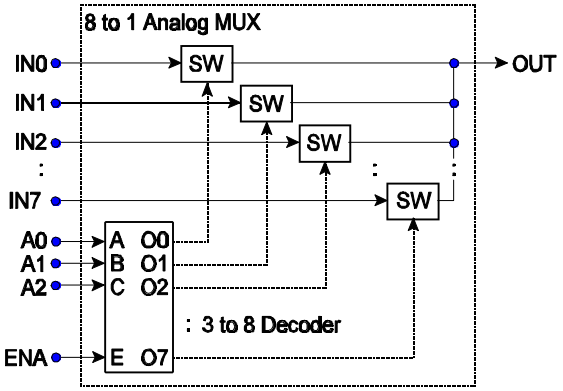


S/W INTERFACE FOR THE SA ADC

- Read from port PSOC, this activates /SOC = '0',
  - Activates /BUSY = '0' ➜ T/S = '0' ➜ SH stores (holds) analog input at V_{IN}.
  - Starts the SA conversion process,
- When conversion finishes,
  - /BUSY changes from '0' to '1' ➜ signals end of conversion, also
  - SH tracks (samples) analog input at V_{IN}.
- For polled handshaking : read port PSTATUS and wait until /BUSY = '1'.
- Read the converted code,
  - Read port, PMSB, for 8-MSB's,
  - Read port, PLSB, for 4-LSB's (left justified).

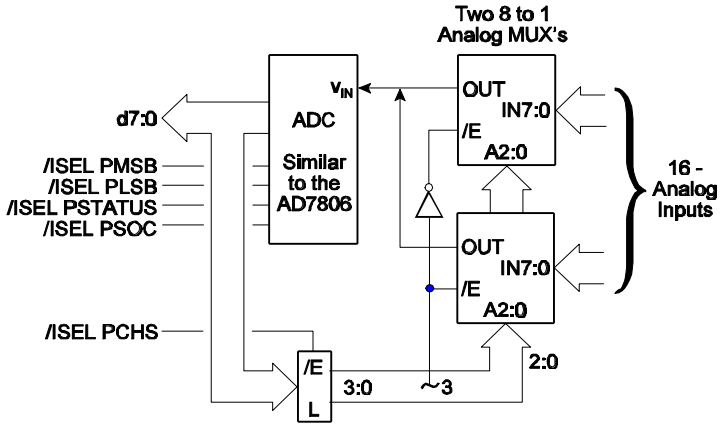
ANALOG MULTIPLEXERS

- If (A2:0 == j and ENA == '1')
- { OUT = the selected analog input, In j ; }
- SW : digital controlled analog switch, '0' open, '1' closed.



MULTI-CHANNEL ANALOG INPUT SYSTEM H/W

- To support multiple analog inputs, the ADC may be preceded by a 2^M to 1 analog multiplexer.



MULTI-CHANNEL ANALOG INPUT SYSTEM S/W

- Operation of the multi-channel analog input system,
  - Write to port PCHS to select the MUX channel,
  - Wait for the MUX switching transients to settle,
  - Read port PSOC to initiate a conversion,
  - Wait for the conversion to finish (may poll /BUSY at port PSTATUS),
  - Read the converted code at ports PLSB and PMSB,
  - Convert code to a (right justified) integer.