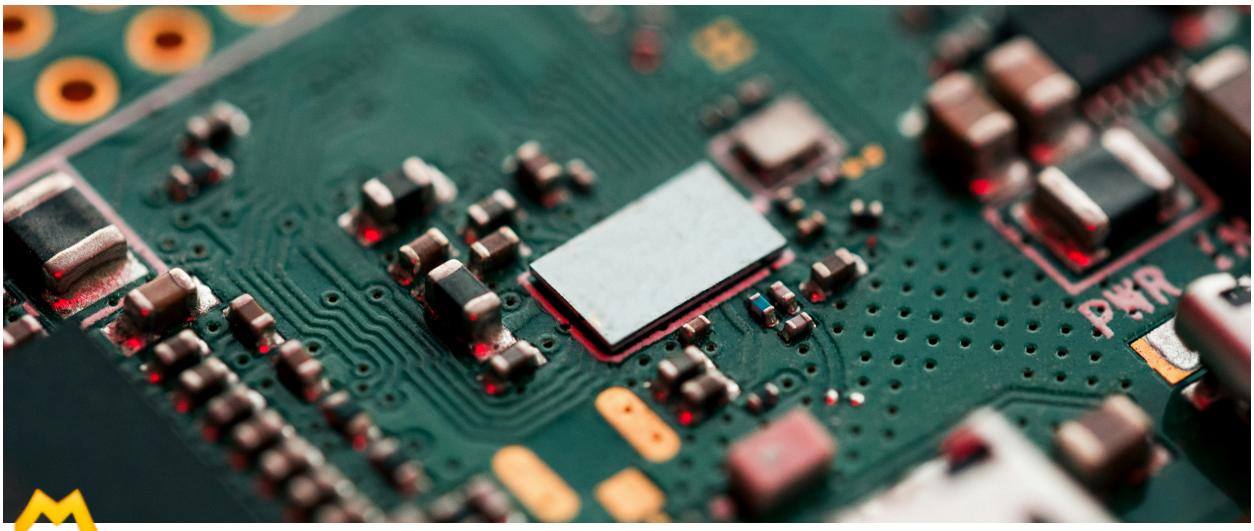


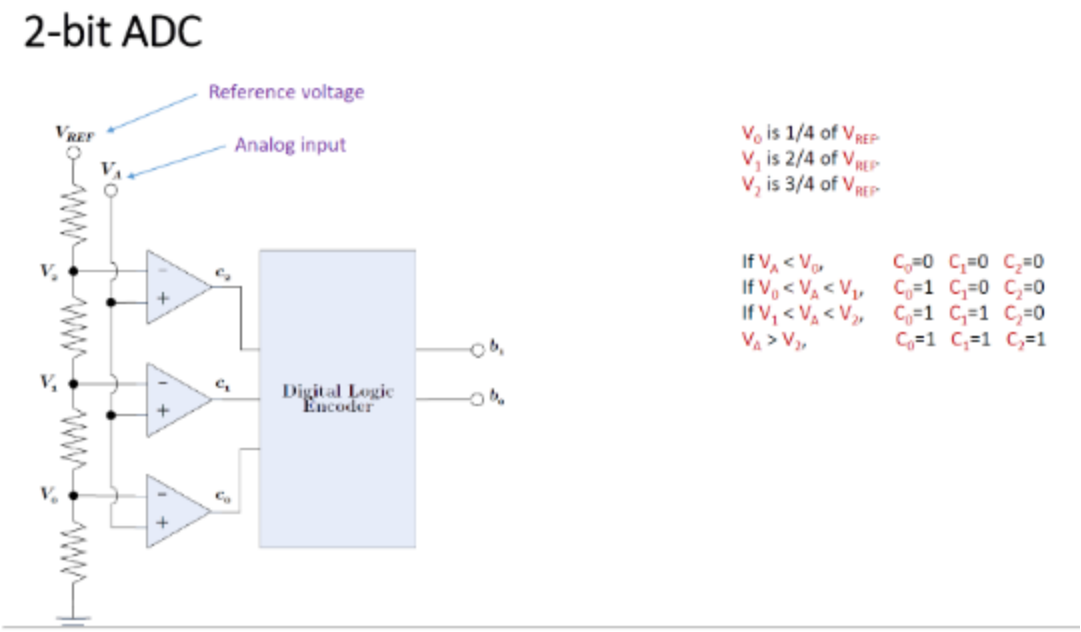
Week 5 - Analog Input

- An analog 0 to 5V voltage can have an infinite number of real-world values
  - Number of different voltage level depends on the number of bits on the ADC
    - Voltage level = 2b , where b = number of bits

ADC based on Combinational Logic

- *Combinational circuit: The output depends only on the present input*
- The voltage divider (made up of equal resistors) splits the V_ref, with highest voltage on top
- The comparators compare the reference voltage to the analog input. If the input is greater than the reference voltage, the comparator outputs 1, otherwise it outputs 0.

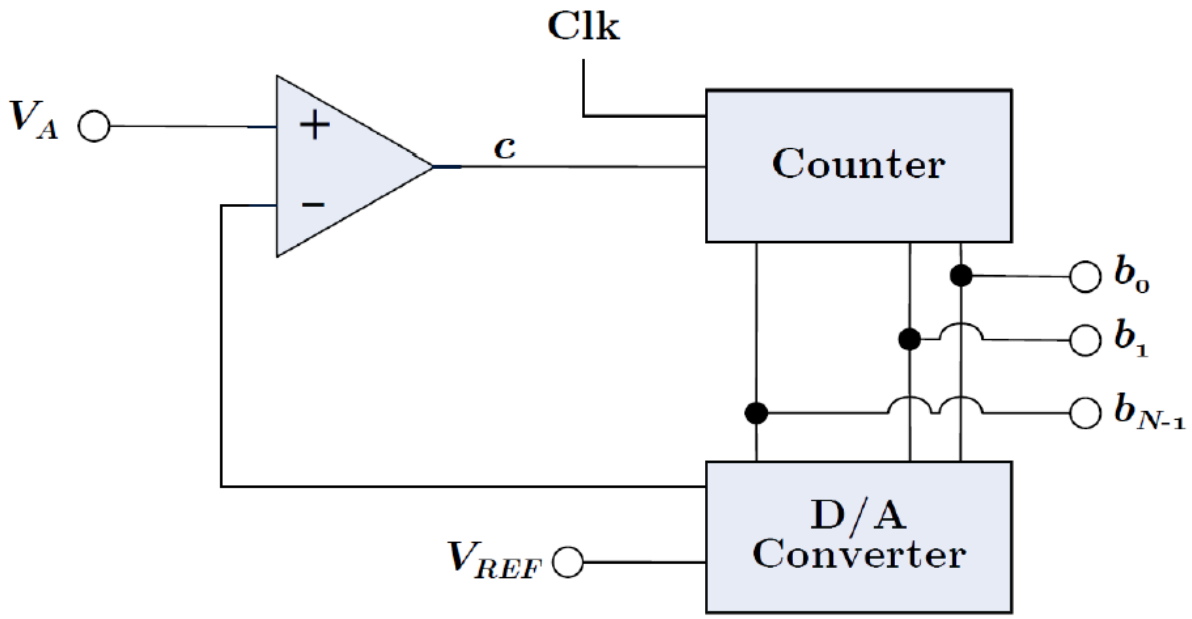
  $2^b - 1$: Number of comparators needed



This is the fastest type of ADC as everything happens in parallel, but it requires many comparators.

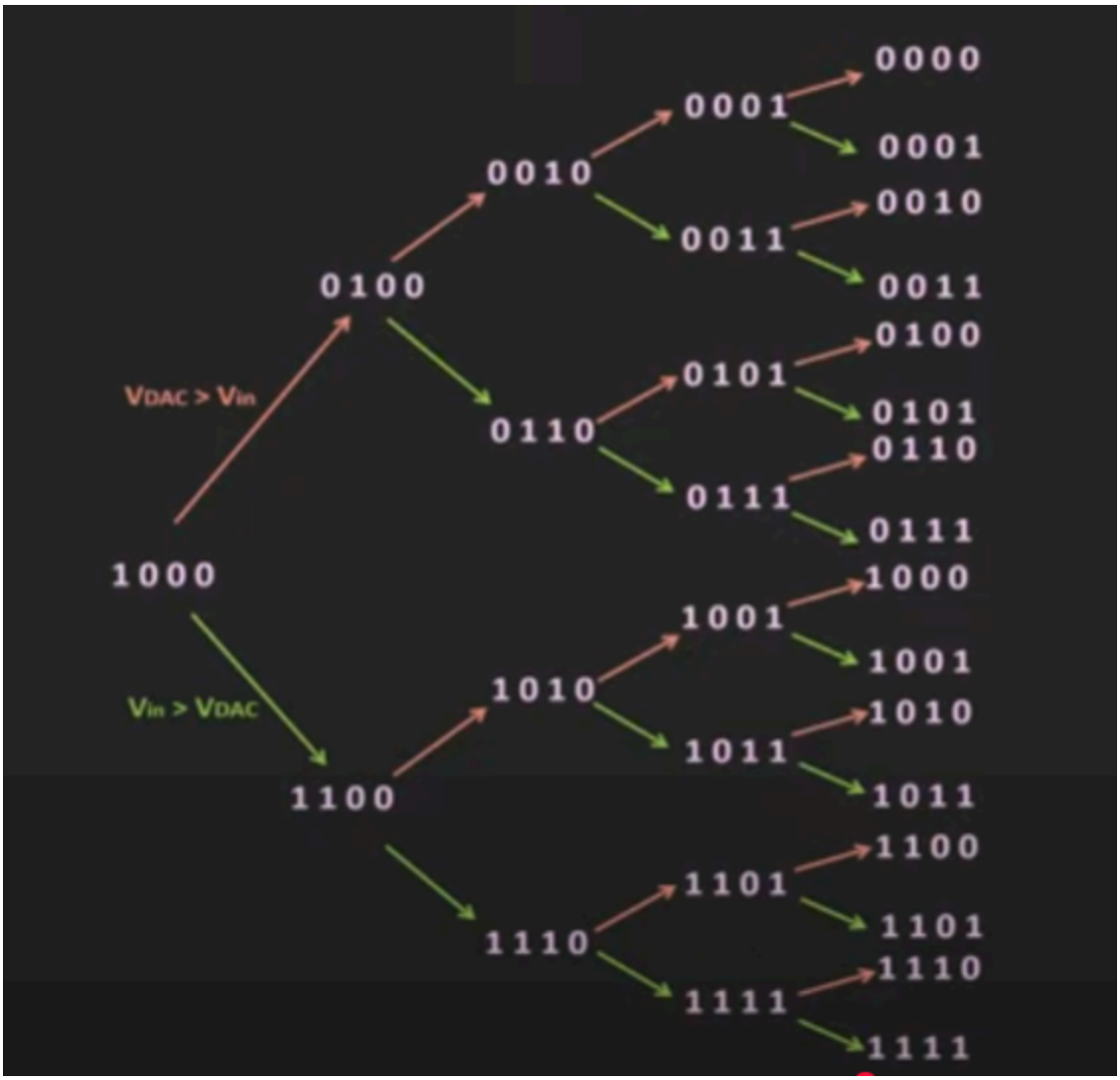
ADC based on Sequential Logic

- *Sequential circuit: The output depends on present and past inputs. (previous output is stored in flip flops)*
- Uses binary search for N+1-clock cycles to determine the discrete voltage level. (N: no. of bits)



Steps:

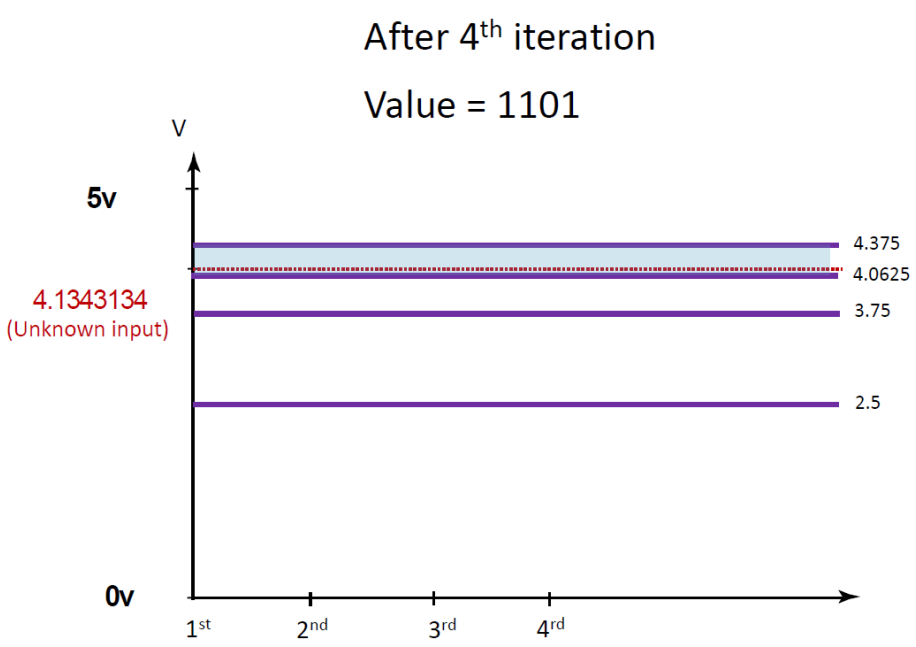
- V_a is the analog signal that is sampled and held.
- Initially, the counter passes 100 (only MSB as high, which is the half value of the ADC) to the DAC which outputs the equivalent analog voltage to the comparator.
- The comparator compares the reference from the DAC and the V__a.
  - If V_a > V_dac: the next significant bit to the DAC becomes 1, i.e. 110
  - If V_a < V_dac: MSB becomes 0, next bit becomes 1, i.e. 010



- Repeat comparison until N clock cycles.

Example:

Determine the digital output value of a 4-bit successive approximation ADC if analog input is 4.1343134 and VREF= 5V.



- Now the number is anywhere between 3.75 and 4.375.
- So the best guess now is 4.0625.
- The value is higher than the guess, so the next bit is 1

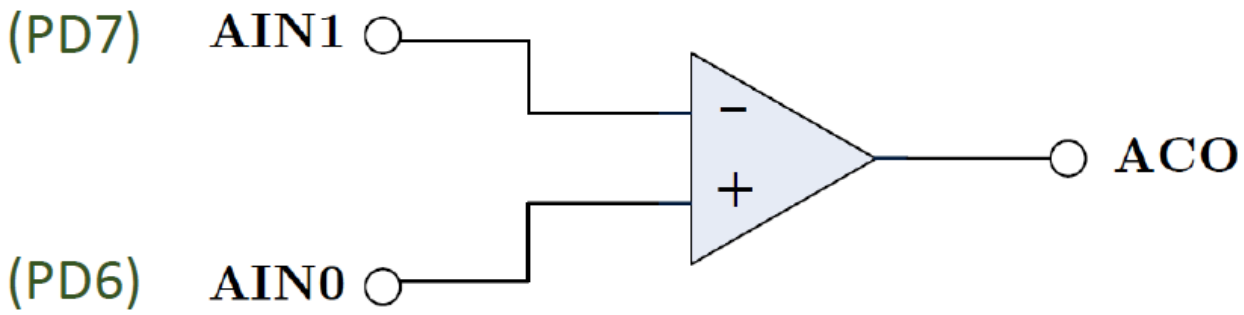
ADC on the ATmega328p

- 10-bit successive approximation ADC. There's only one ADC engine on the ATmega328p, even though it has six input channels (PC0–PC5)
- Max frequency: **1 MHz**
- Takes 13 - clock cycles

1. **Sample & Hold Acquisition (1 cycle):**
   The internal S/H capacitor must charge up to the input voltage.
2. **Successive-Approximation Bit Trials (10 cycles):**
   One compare-and-set operation per bit, MSB→LSB.
3. **Result Register Update & Overhead (2 cycles):**
   Finalizes the conversion, latches the 10-bit result into ADCL/ADCH.

So 1 + 10 + 2 = **13 ADC clocks.**

- If you wire three sensors to A0, A1, A2, you're *not* sampling them simultaneously.
- You sample them *sequentially* (depending on which order analogRead() is called in code), at whatever overall rate you can afford. After one pin has been sample, the sampler is swtiched to another pin using a MUX.
- The ATmega328p has one built in analog comparator. This comparator can be used for simple thresholding of analog values (by also adding resistors to get different voltage levels for comparison). It saves power and is faster as its instant and doesnt need any clock cycles.



Two Modes of ADC

On-Demand Mode:

- Auto-trigger disabled (ADATE = 0 in register)
- Arduino uses this [analogRead()]
- User decides when to start sampling

Auto - Trigger Mode

- Auto - trigger enabled (ADATE=1)
- As soon as one conversion gets completed, another conversion automatically gets carried out
- Works best if need to measure only one channel.

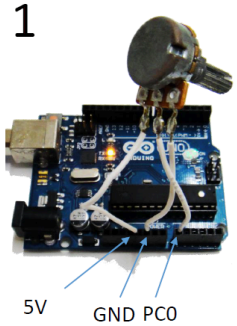
Raspberry Pi

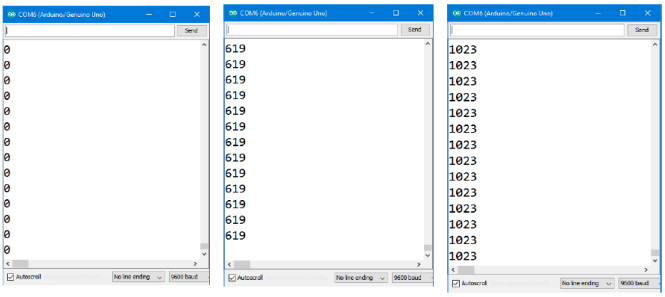
- Has no built-in ADC. Need to use an external SAR ADC (e.g. MCP3008) over SPI. Sequential logic.
  - With MCP3008 (10-bit) and Pi 3's 3.3 V rail → ≈3.3 V/1023 ≈ 3.225 mV per LSB.
- SPI (Serial Peripheral Interface) on the Raspberry Pi is a hardware-supported, full-duplex, synchronous serial bus you use to talk to chips like ADCs, DACs, sensors, displays, etc.
  1. **How it works**
  - **Master/Slave:** Pi is the master; each peripheral (e.g. MCP3008) is a slave.
  - **4 wires:**
    - **SCLK (Serial Clock):** Pi-generated clock.
    - **MOSI (Master Out, Slave In):** Data from Pi → device.
    - **MISO (Master In, Slave Out):** Data from device → Pi.
    - **CE/CS (Chip Enable/Select):** Pi pulls low to talk to one device at a time.

▼ Exercises

Example 1

The program prints out values from 0 to 1023 (according to the position of the potentiometer)






```
int main()              //To run on Arduino, just change this function to void setup()
{
    Serial.begin(9600);   //For now, we are still using the Serial library

    for (;;)
    {
        int value = analogRead(A0); //int or unsigned int does not matter
        Serial.println(value);
    }
}
```

Example 2

Write a program that continuously prints out the analog value of PC3 through the serial port. Assume the followings:

- The clock speed of the microcontroller is 16MHz
- The desired clock speed of ADC is 125 kHz.
- Use AVCC as analog reference.
- On-demand mode

Example 3

Modify example 2 to put the ADC in continuously running mode (auto-trigger)