Digital Logic Design

Lecture 10

Practical considerations for digital design

Detecting the change of a signal

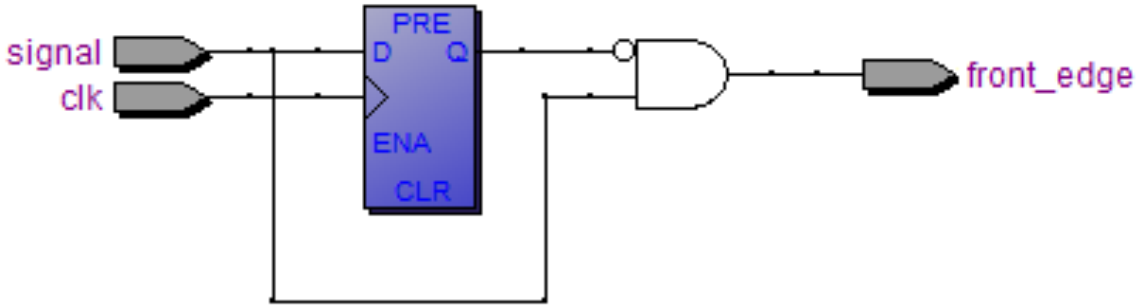
Let's assume we need to check when a (long) `signal` changes.

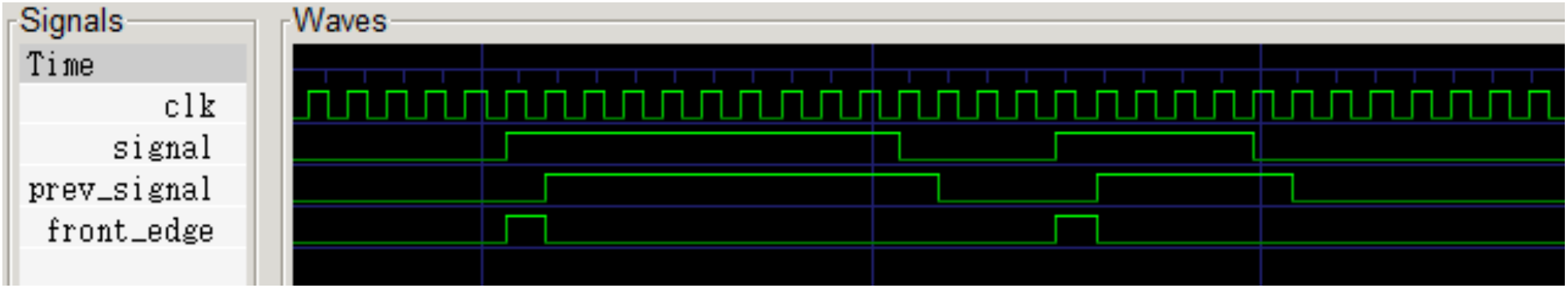
Store the `signal` into the register `prev_signal`

Compare the current value of the `signal` with its previous value `prev_signal`

Rising edge detection

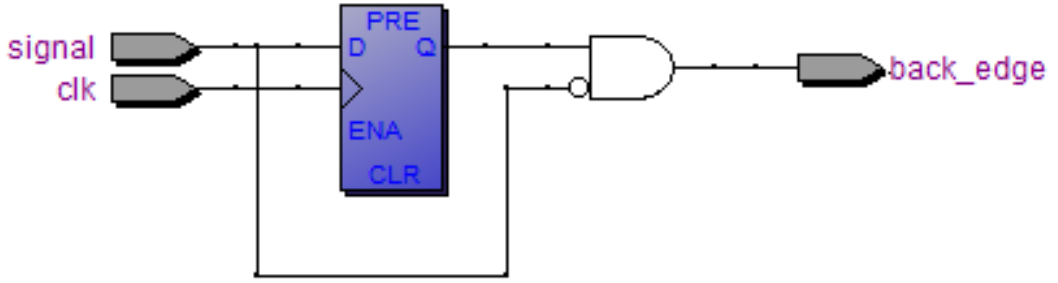
```
reg prev_signal;

always @(posedge clk)
  prev_signal <= signal;

wire front_edge;
assign front_edge = ~prev_signal & signal;
```

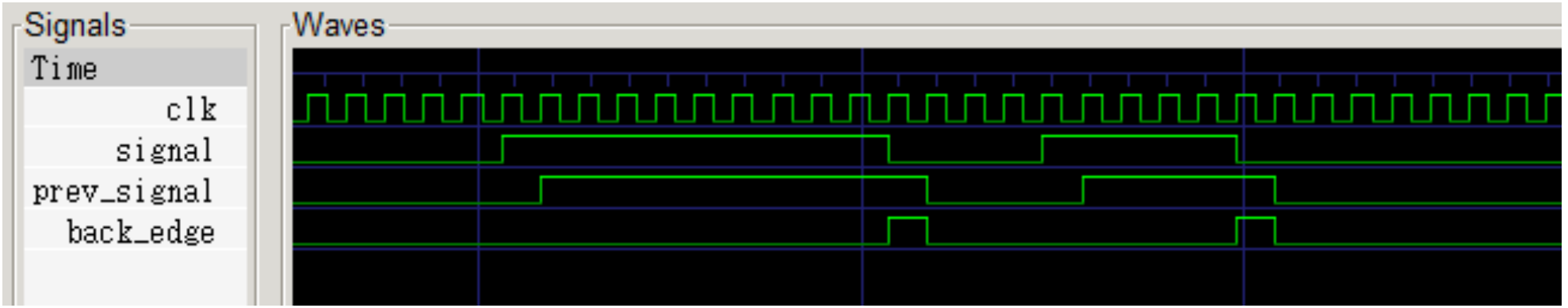




Falling edge detection

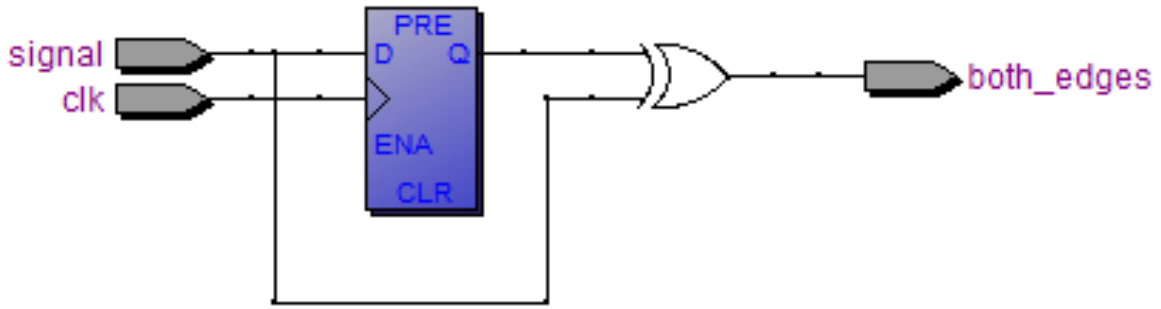
```
reg prev_signal;

always @(posedge clk)
  prev_signal <= signal;

wire back_edge;
assign back_edge  =  prev_signal & ~signal;
```

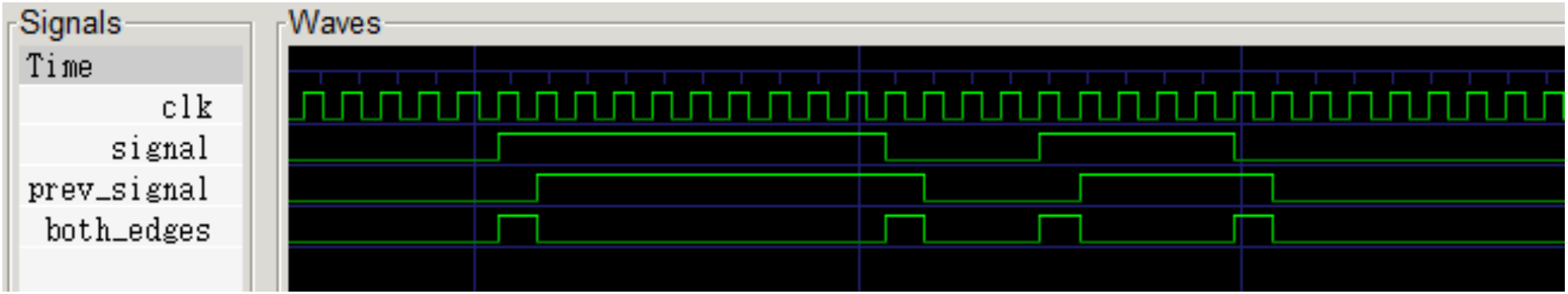




Rising and falling edge detection

```
reg prev_signal;

always @(posedge clk)
  prev_signal <= signal;

wire both_edges;
assign both_edges =  prev_signal ^ signal;
```





Schmitt trigger

A **Schmitt trigger** is a circuit that is used to transform slowly changing waveforms into sharply defined, jitter-free output signals.

They are useful for changing clock edges that may have slow rise and fall times into straight vertical edges.

Most CMOS, BiCMOS and TTL devices require fast edges on the high and low transitions on their inputs.

If the edges are slow they can cause excessive current, oscillation and even damage the device.

Slow edges are hard to avoid at power up or when using push button or manual type switches with the large capacitors needed for filtering.

On a non-Schmitt trigger input the part will switch at the same point on the rising edge and falling edge.

????

With a slow rising edge the part will switch at the threshold.

When the switch occurs it will require current from V_{CC}

When current is forced from V_{CC} , the V_{CC} level can drop causing the threshold to shift.

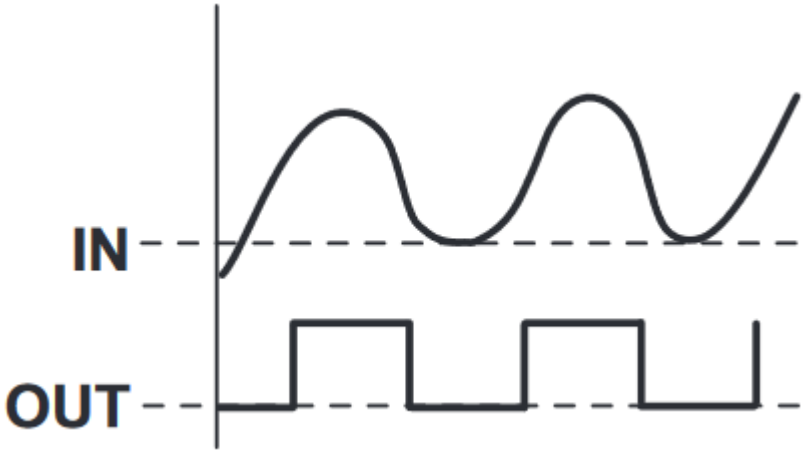
When the threshold shifts it will cross the input again causing the part to switch again.

This can go on and on causing oscillation which can cause excessive current.

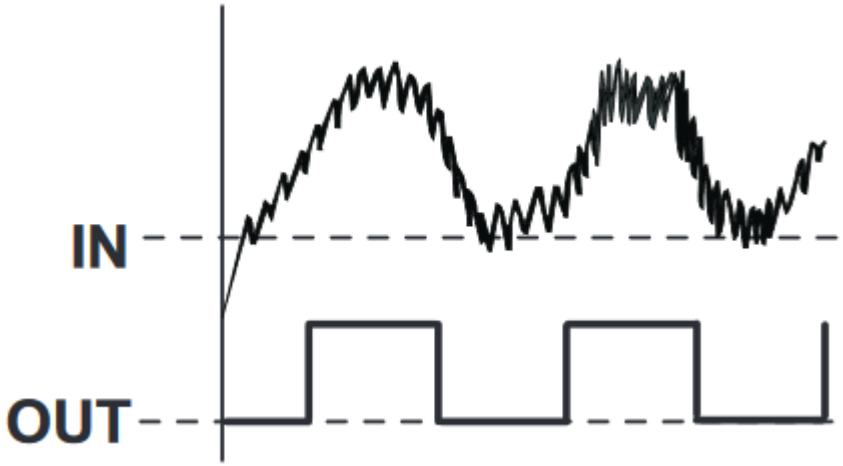
The same thing can happen if there is noise on the input.

Schmitt triggers should be used any time you need to

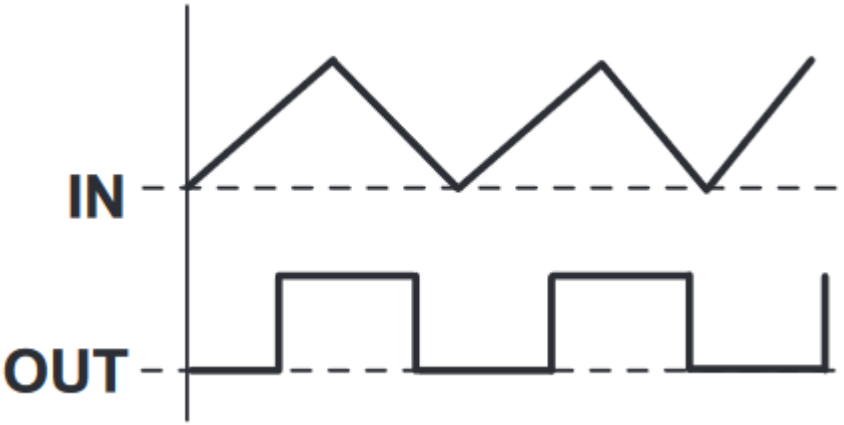
– change a sine wave into a square wave



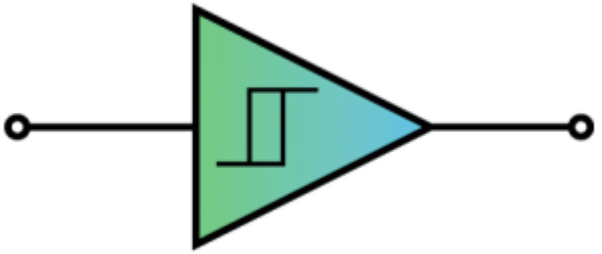
– clean up noisy signals



– convert slow edges to fast edges

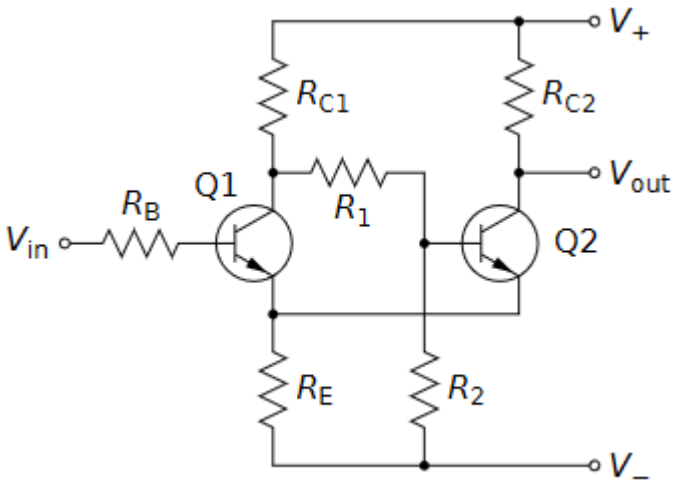


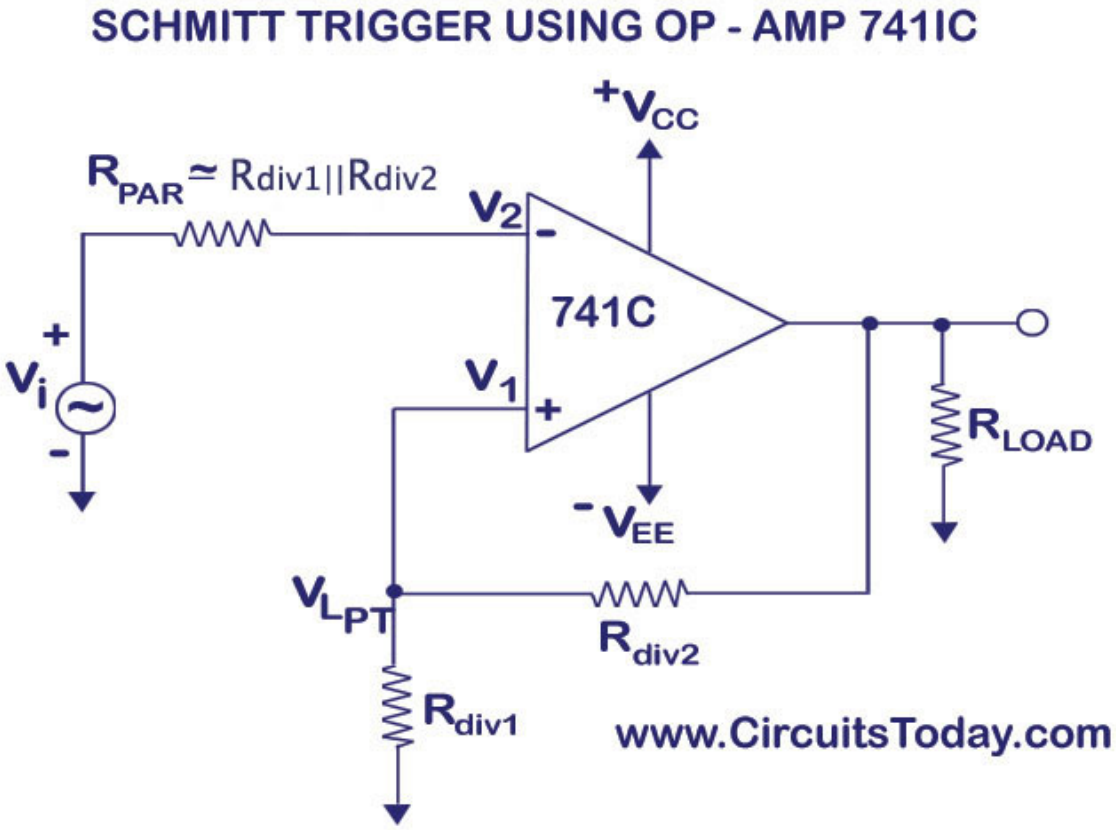
Circuit symbol for a Schmitt trigger



Schmitt Triggers can be built using transistors or an Operational Amplifier

Schmitt trigger implemented by two emitter-coupled transistor stages

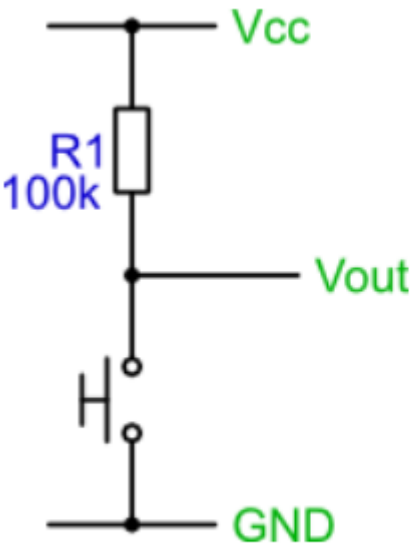




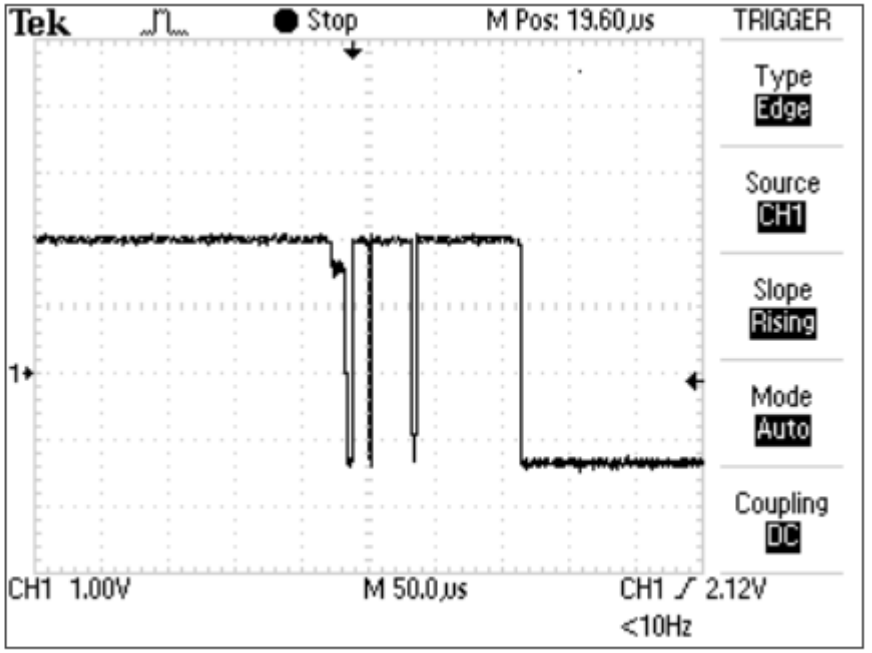
Switch debouncing

If you want to input a manual switch signal into a digital circuit you'll need to **debounce** the signal so a single press doesn't appear like multiple presses.

The contacts within the switch don't make contact cleanly, but actually slightly 'bounce'.



Simple switch pull-up circuit



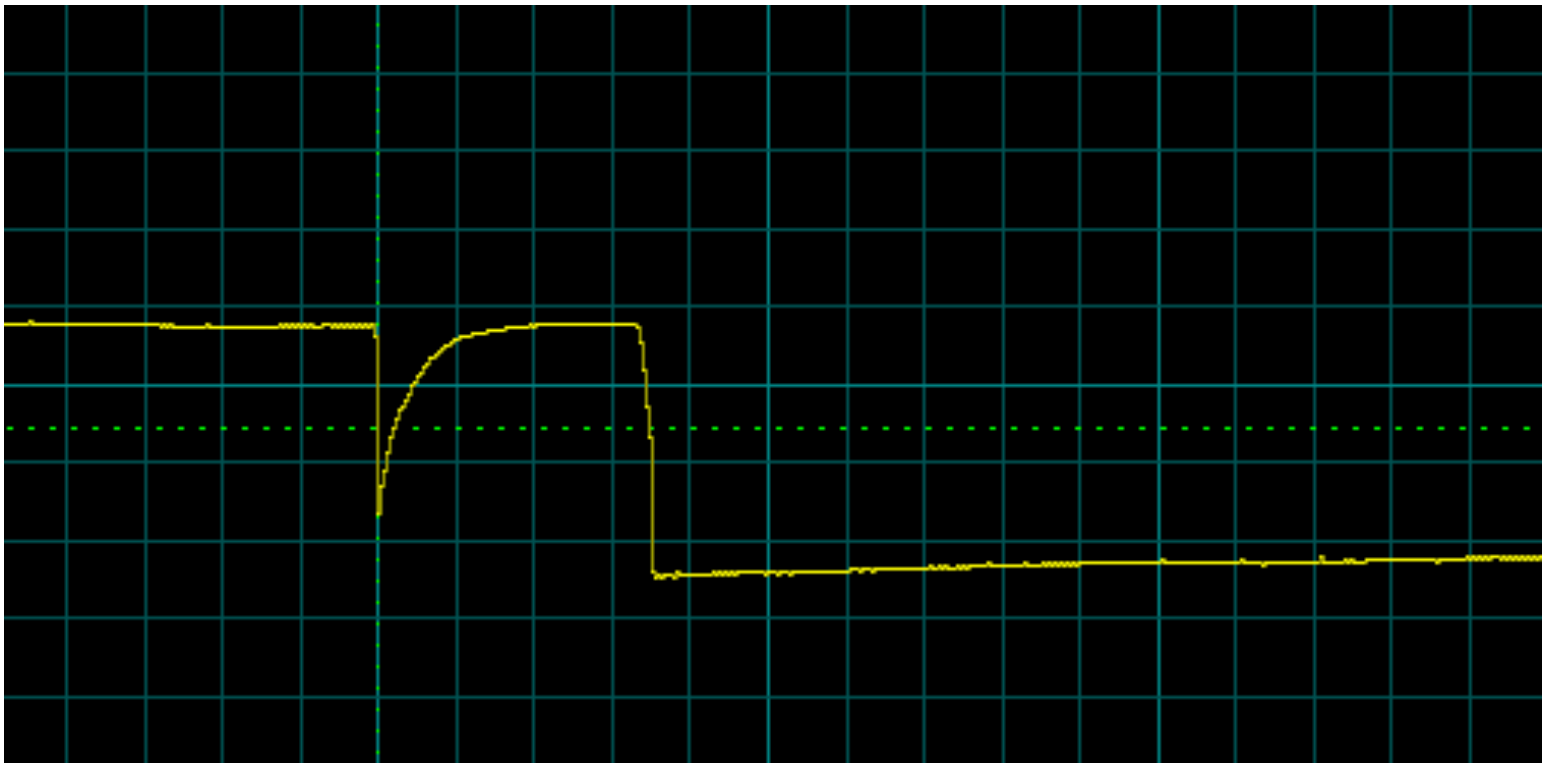
Switch bounce produced on switch press

The same can also occur on the release fo a switch.

When the contacts of a switch are closed, the electrical and mechanical connection is first made, but due to a slight springing action of the contacts, they will bounce back open, then close, then open, then close, continuing repeatedly until they finally settle down in the closed position.

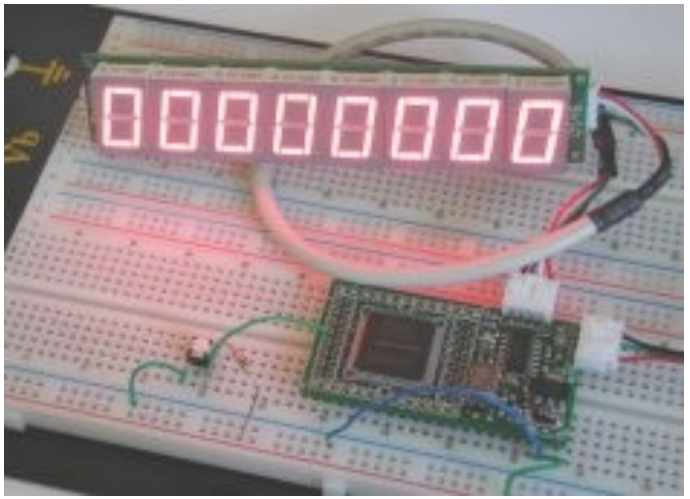
This bouncing action will typically take place for as long as 50 ms.

Example:

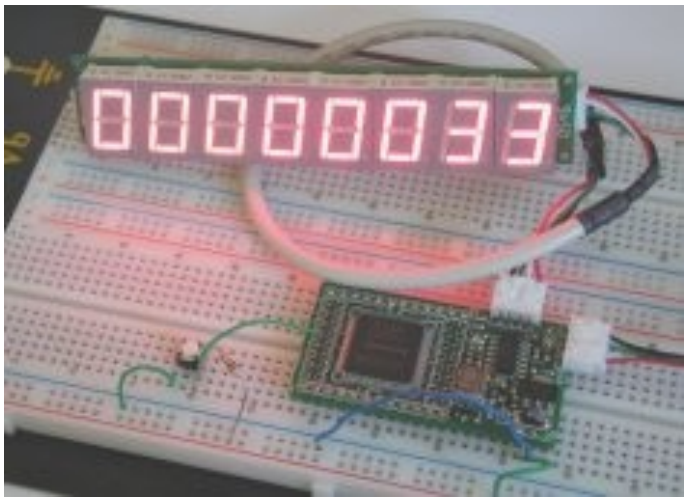


Example: FPGA counter

Power-on, things look good.

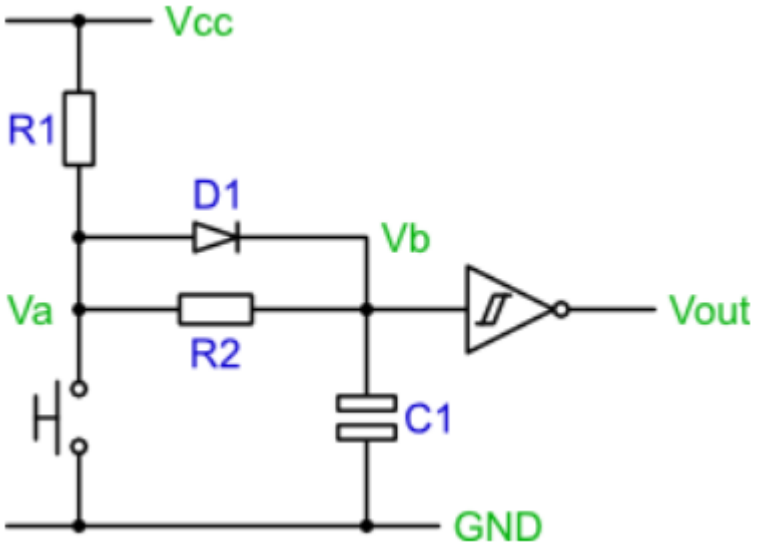


You press the push-button 10 times, and...



A switch debouncer circuit

The basic idea is to use a capacitor to filter out any quick changes in the switch signal.



When the switch is open

C1 will charge via R1 and D1

In time, C1 will charge and Vb will reach within 0.7V of Vcc.

Therefore the output of the inverting Schmitt trigger will be a logic 0.

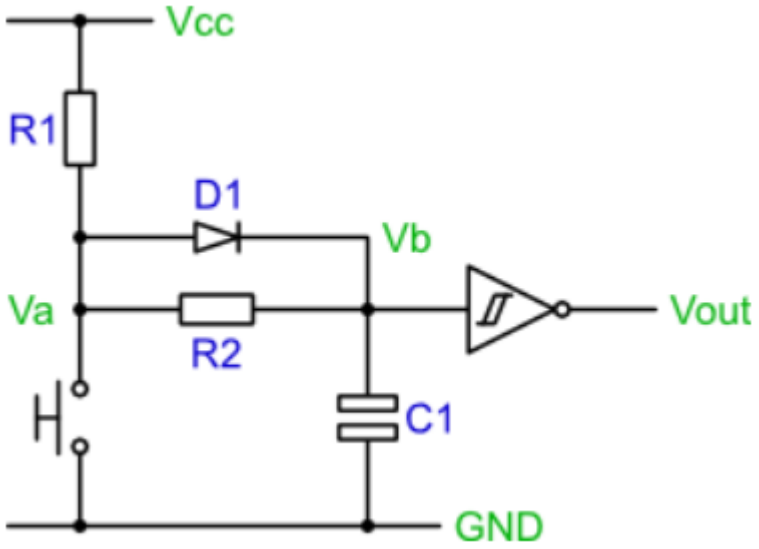
Now close the switch

C1 will discharge via R2

In time, C1 will discharge and Vb will reach 0V.

The output of the inverting Schmitt trigger will be a logic 1.

If bounce occurs and there are short periods of switch closure or opening, the capacitor will stop the voltage at Vb immediately reaching Vcc or GND.



Although, bouncing will cause slight charging and discharging of the capacitor, the Schmitt trigger will stop the output from switching.

R2 is required as a discharge path for the capacitor.

Without it, C1 will be shorted when the switch is closed.

Without the diode, D1, both R1 and R2 would form the capacitor charge path when the switch is open.

The combination of R1 and R2 would increase the capacitor charge time, slowing down the circuit.

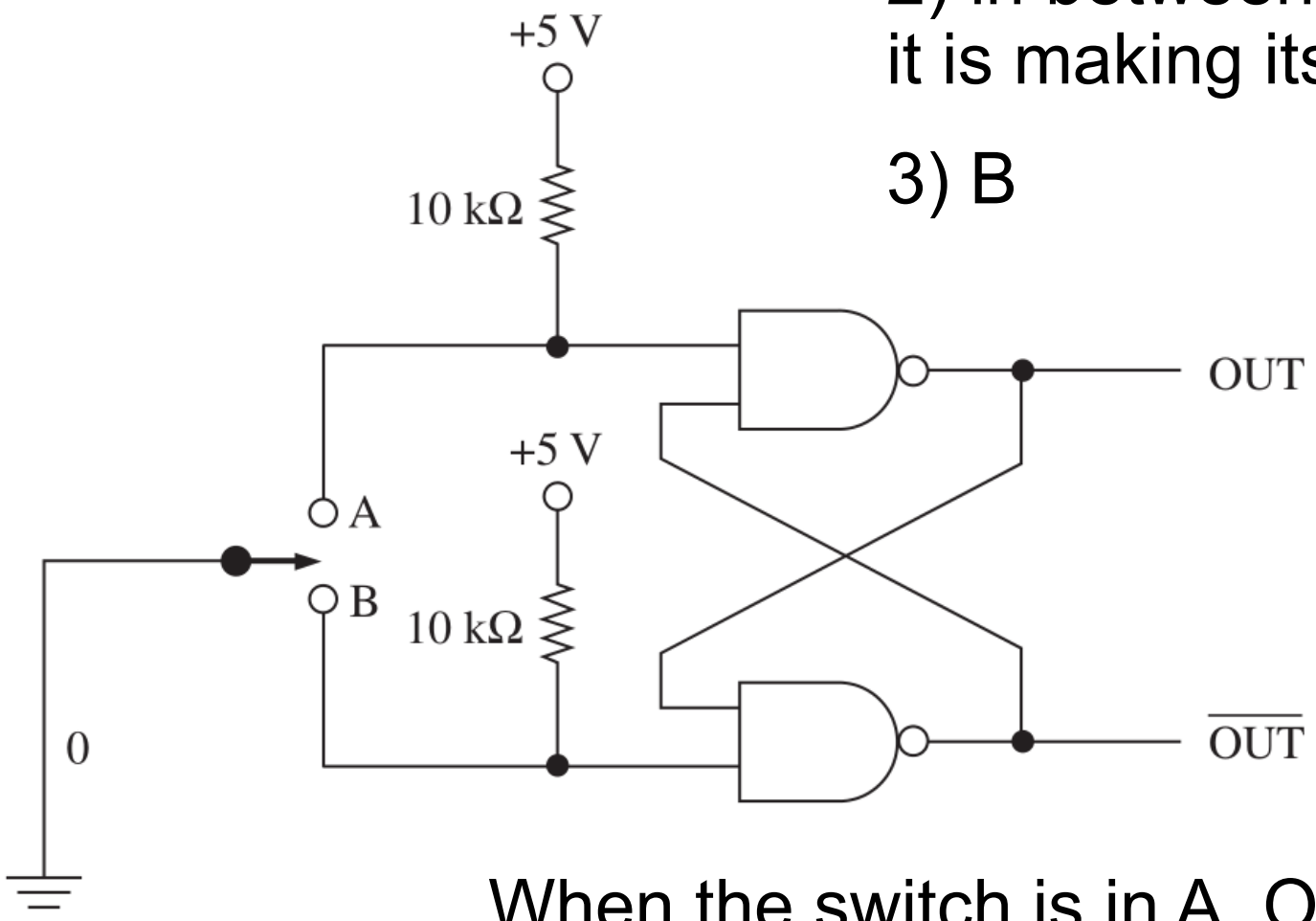
Debouncing a single-pole, double-throw switch

3 positions of the switch:

1) A

2) in between A and B while it is making its transition

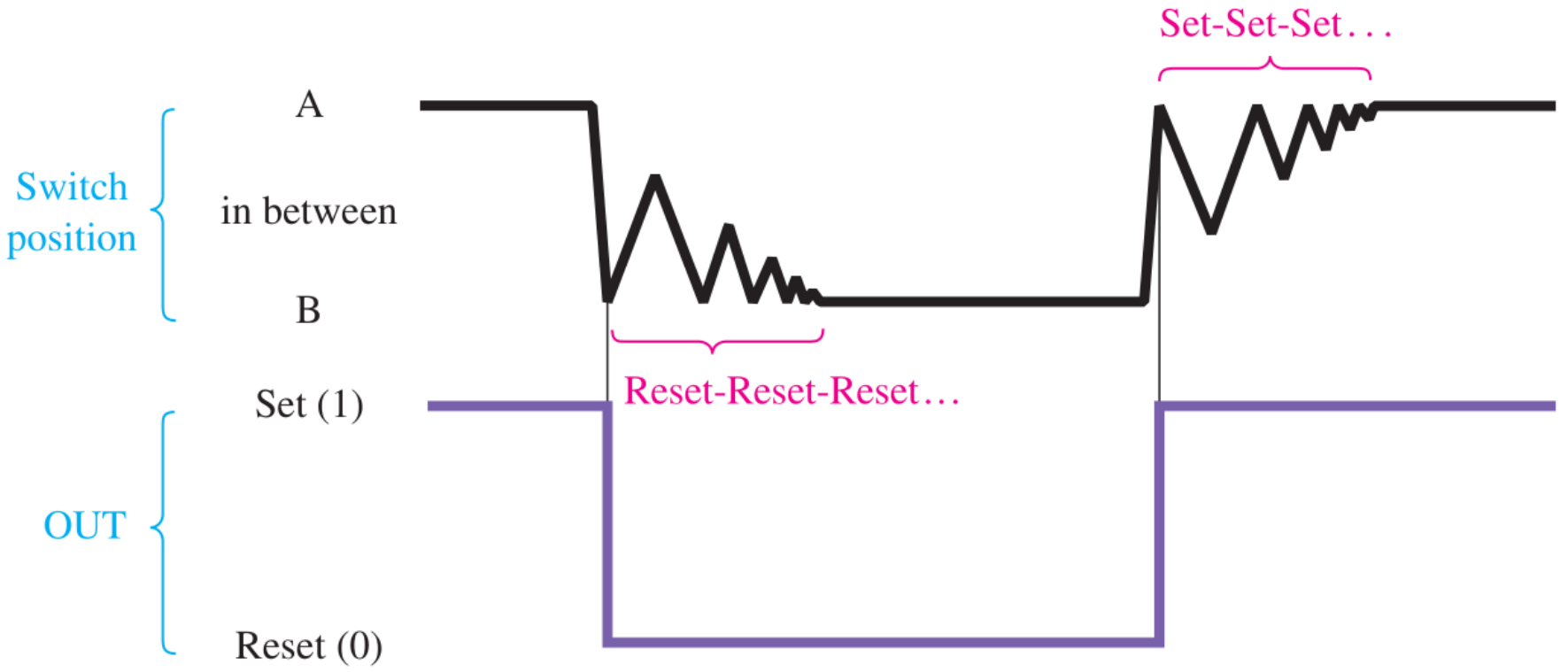
3) B



When the switch is in A, OUT will be set (1)

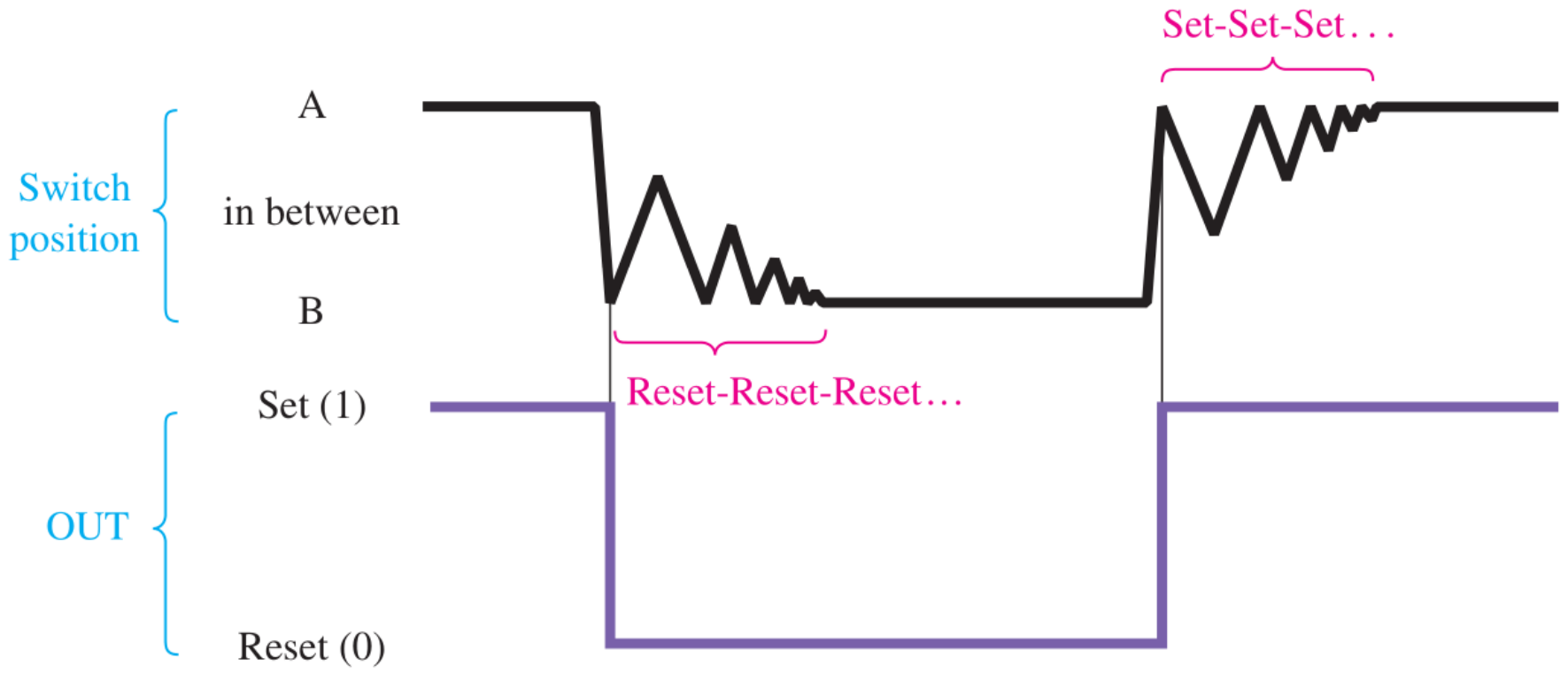
When the switch is moved from A to B, it bounces, causing OUT to Reset, Hold, Reset, Hold, Reset, Hold repeatedly until the switch stops bouncing and settles to B

From the time the switch first touched B until it is returned to A, OUT will be Reset even though the switch is bouncing.

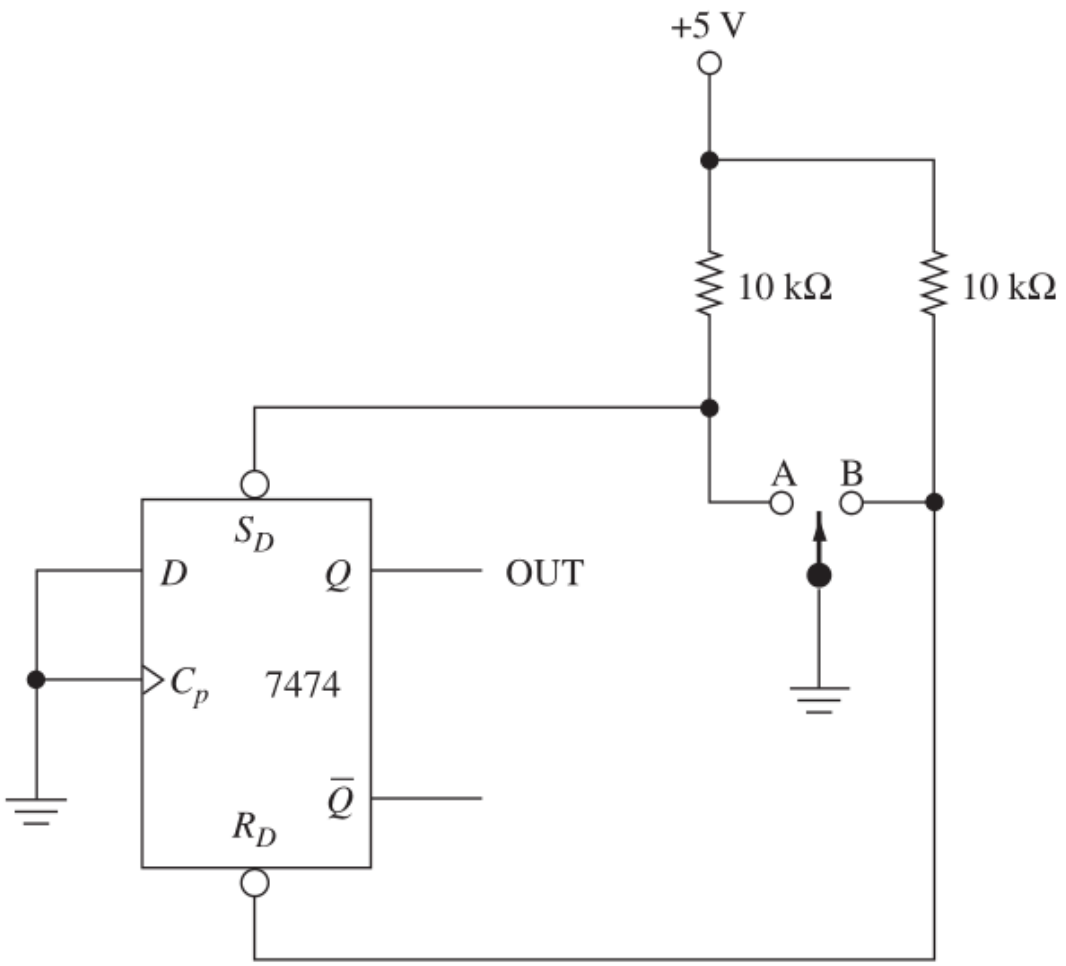


When the switch is returned to A from B, it will bounce, causing OUT to be Set, Hold, Set, Hold, Set, Hold repeatedly until the switch stops bouncing.

In this case, OUT will be Set and remain Set from the moment the switch first touched A, even though the switch is bouncing.



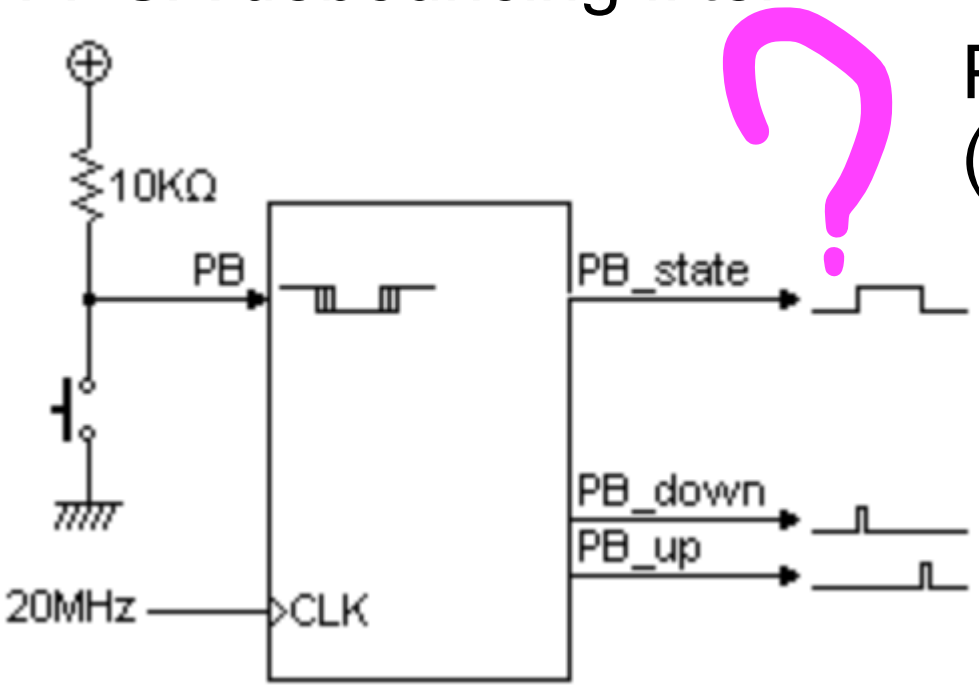
Debouncing a single-pole, double-throw switch using a D flip-flop



As the switch is moved to A but is still bouncing, the flip-flop will Set, Hold, Set, Hold, Set, Hold repeatedly until the switch settles into A, keeping the flip-flop Set.

When it is moved to B, the flip-flop will Reset, Hold, Reset, Hold, and so on until it settles down, keeping the flip-flop Reset.

FPGA debouncing filter



PB is the push-button signal (active low in this example).

It may contain glitches, and is asynchronous to any clock.

So it is unusable.

We are going to synchronize PB to a clock (20MHz in this example).

Then we create three push-buttons outputs, glitch free, synchronous to the clock.

Each output will be active high and indicate a different condition of the push-button (push-button state, just pushed, just released).

```
module PushButton_Debouncer(
  input clk,

  // "PB" is the glitchy, asynchronous to clk, active low push-button signal
  input PB,

  // from which we make three outputs, all synchronous to the clock
  output reg PB_state,  // 1 as long as the push-button is active (down)
  output PB_down,  // 1 for one clock cycle when the push-button goes down
  output PB_up   // 1 for one clock cycle when the push-button goes up
);

  // First use two flip-flops to sync the PB signal the "clk" clock domain
  // also invert PB to make PB_sync_0 active high
  reg PB_sync_0;  always @(posedge clk) PB_sync_0 <= ~PB;
  reg PB_sync_1;  always @(posedge clk) PB_sync_1 <= PB_sync_0;

  reg [15:0] PB_cnt; // Declare a 16-bits counter

  // When the push-button is pushed or released, we increment the counter
  // The counter has to be maxed out before we decide that the push-button state
  // has changed

  wire PB_idle = (PB_state==PB_sync_1);
  wire PB_cnt_max = &PB_cnt;    // true when all bits of PB_cnt are 1's
```

Cont. on the next page

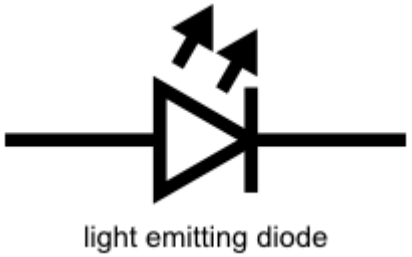
```
  always @(posedge clk)
  if(PB_idle)
    PB_cnt <= 0;  // nothing's going on
  else
  begin
    PB_cnt <= PB_cnt + 16'b1;  // something's going on, increment the counter
    // if the counter is maxed out, PB changed!
    if(PB_cnt_max) PB_state <= ~PB_state;
  end

  assign PB_down = ~PB_idle & PB_cnt_max & ~PB_state;
  assign PB_up   = ~PB_idle & PB_cnt_max &  PB_state;
endmodule
```

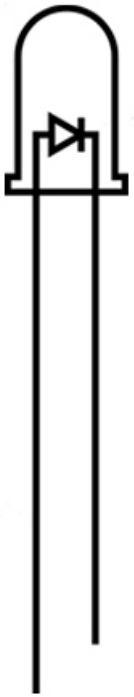
We used a 16-bits counter.

With a 20MHz system clock, it would take 3ms to max-out.

Depending on how glitchy your push-button is and your system clock speed, you might need to adjust the counter width.

Working with Light-Emitting Diodes (LEDs)





The positive side of the LED is called the **anode**

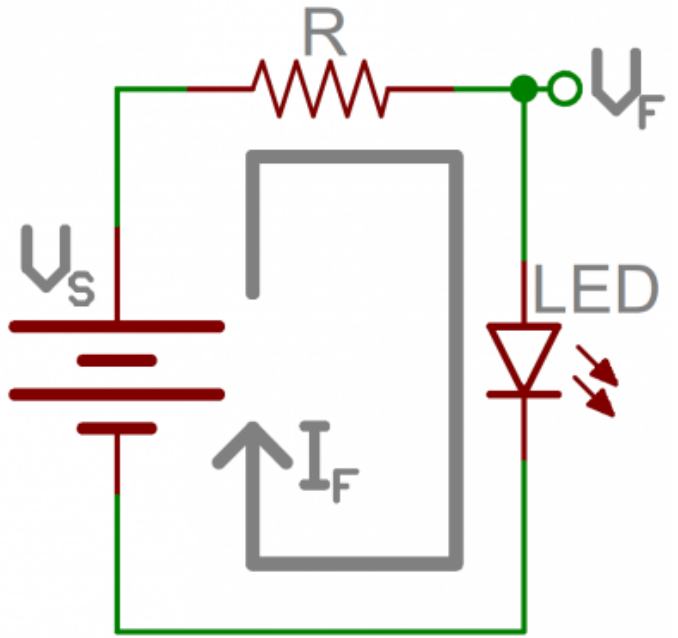
The negative side of the LED is called the **cathode**

Current flows from the anode to the cathode.

The brightness of an LED is directly dependent on how much current it draws.

If you connect an LED directly to a current source it will try to dissipate as much power as it's allowed to draw, and may destroy itself if the current is too large.

It's important to limit the amount of current flowing across the LED.



For this, we employ resistors.

When sizing out a current-limiting resistor, look for two characteristic values of the LED:

1) the typical forward voltage

2) the maximum forward current

The **typical forward voltage** is the voltage which is required to make an LED light up.

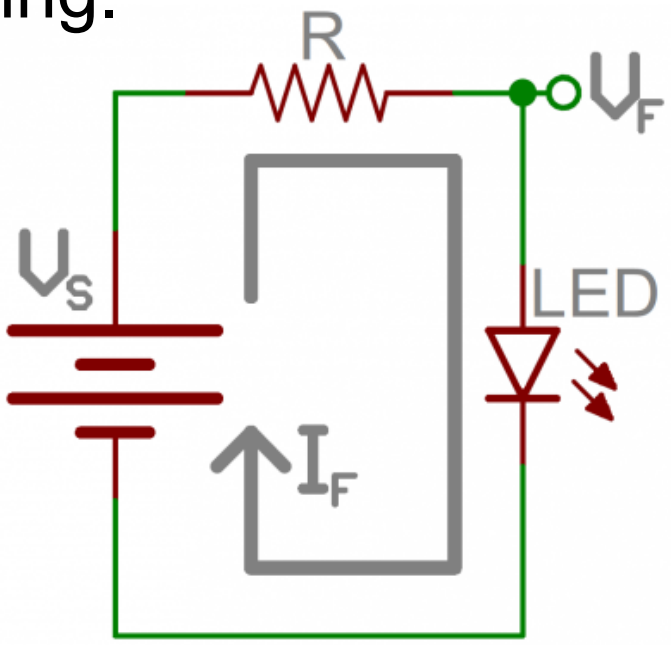
It varies (usually somewhere between 1.7V and 3.4V) depending upon the color of the LED.

The **maximum forward current** is usually around 20mA

Continuous current through the LED should always be equal to or less than that current rating.

The current-limiting resistor is calculated from the equation:

$$R = \frac{V_S - V_F}{I_F}$$



V_S – the source voltage

V_F – the LED's forward voltage

I_F – the desired current through LED

Turning an LED on and off

```
module LEDblink(clk, LED);
  input clk;     // clock typically from 10MHz to 50MHz
  output LED;

  // create a binary counter
  reg [31:0] cnt;
  always @(posedge clk) cnt <= cnt+1;

  // blink the LED at a few Hz
  // using the 23th bit of the counter,
  // use a different bit to modify the blinking rate
  assign LED = cnt[22];
endmodule
```

Making an LED half-lit: drive the LED FPGA output half of the time.

If that's done fast enough, your eye will see the LED half-lit.

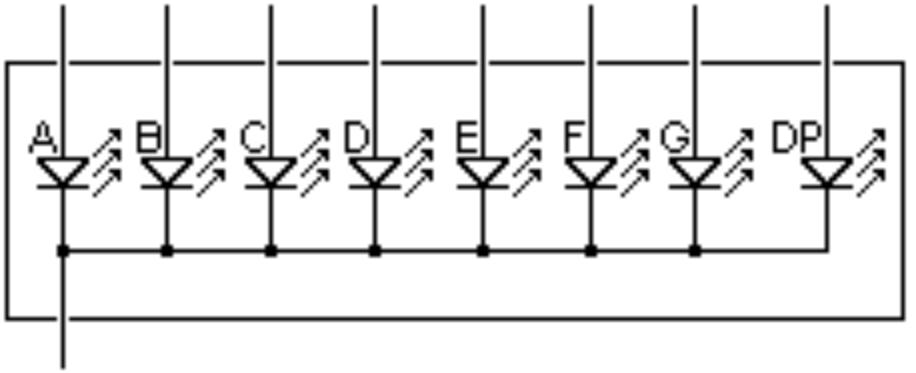
Anything faster than 100Hz is too fast for the human eye to detect - it averages the light intensity it receives.

```
module LEDhalflit(clk, LED);
  input clk;      // clk should be at least 200Hz.
                  // most FPGA boards have clocks,
                  // running at a few 10's of MHz
  output LED;

  reg toggle;
  // toggles at half the clk frequency (at least 100Hz)
  always @(posedge clk) toggle <= ~toggle;

  assign LED = toggle;
endmodule
```

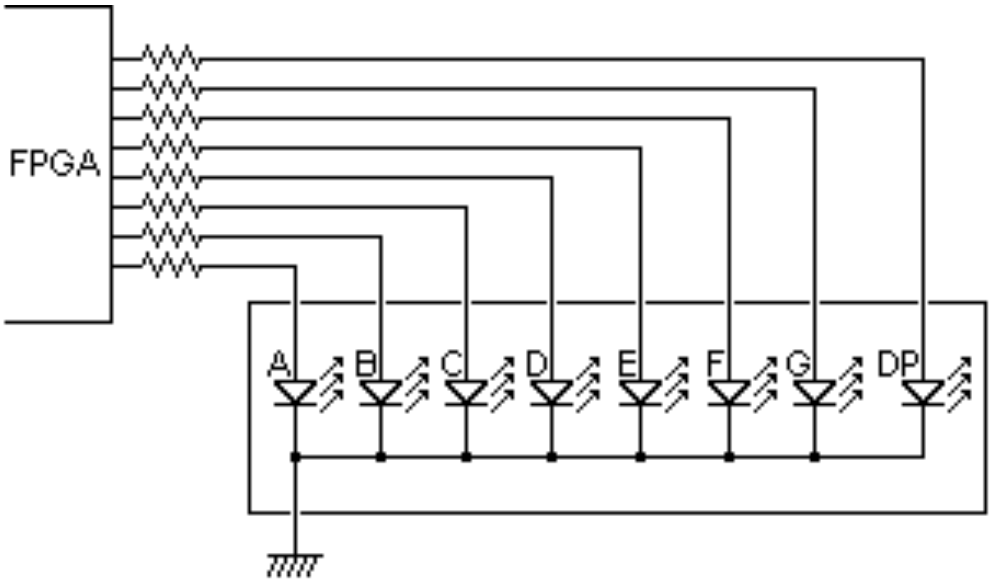
7-segments LED displays

Each segment is created using a separate LED

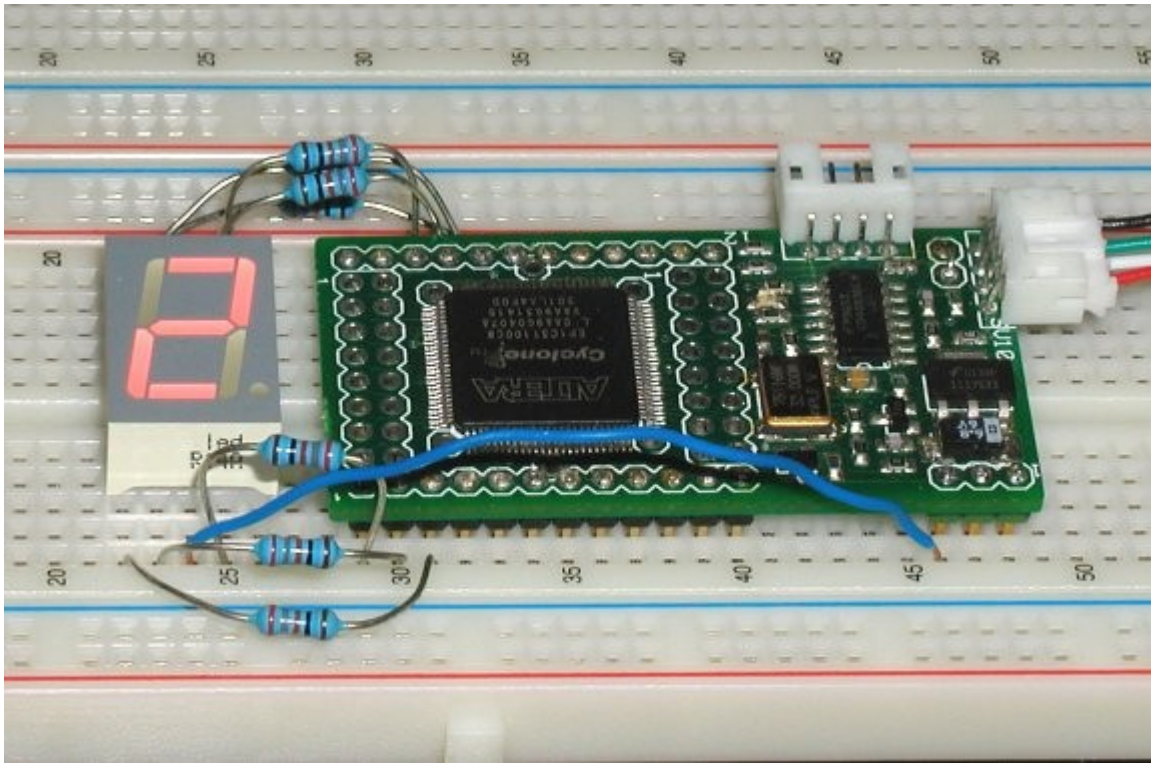
A typical **common-cathode** display:



Common-anode displays have anodes connected together.

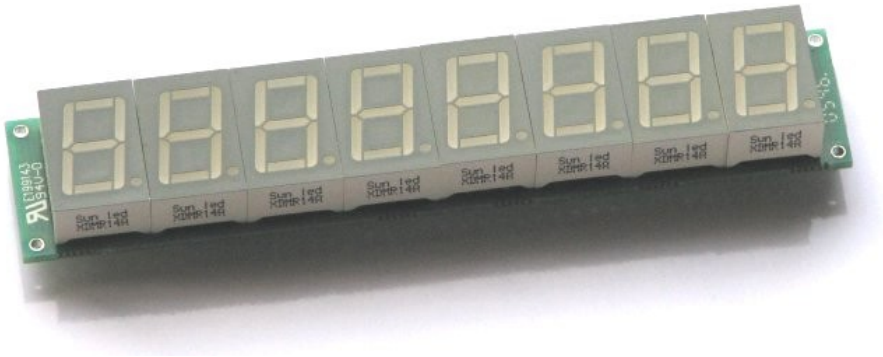


Such displays require at least 9 pins.



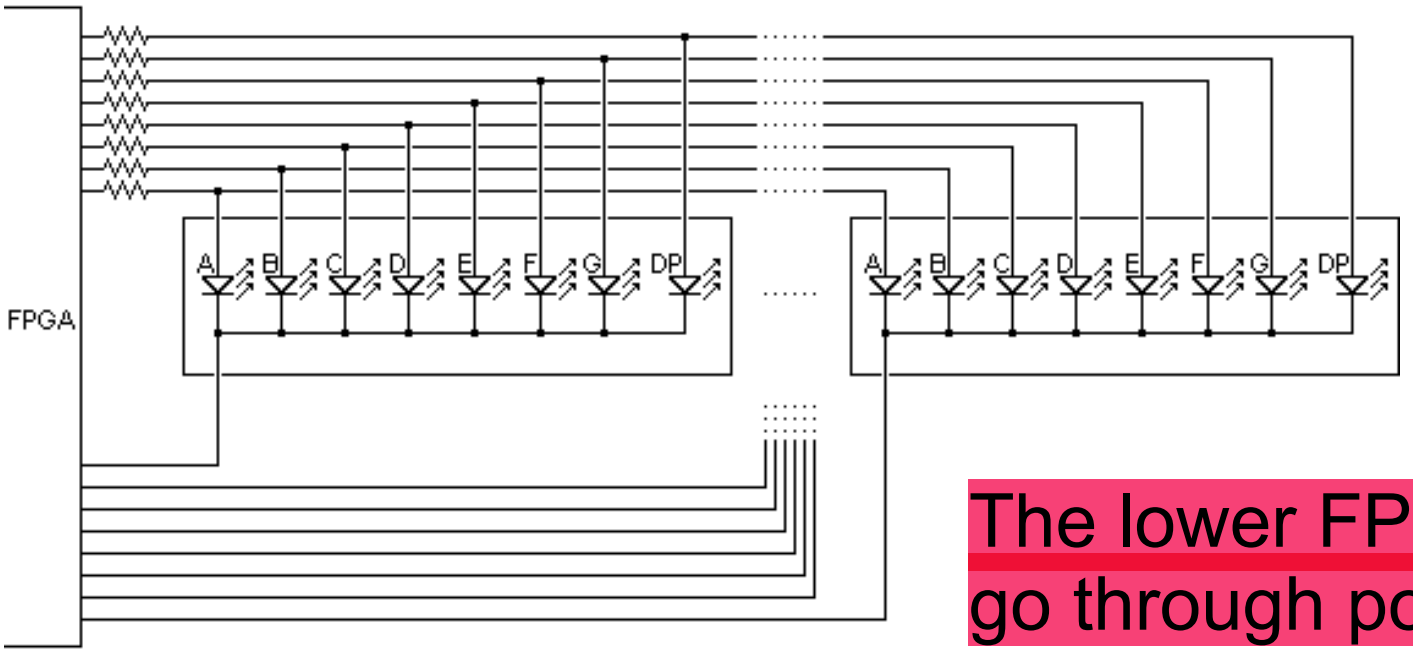

```
module LED_7seg(segA, segB, segC, segD, segE, segF, segG, segDP);
  output segA, segB, segC, segD, segE, segF, segG, segDP;

  // light the leds to display '2'
  assign {segA, segB, segC, segD, segE, segF, segG, segDP} = 8'b11011010;
endmodule
```

LED multiplexing



Only one display can be lit at any particular time.

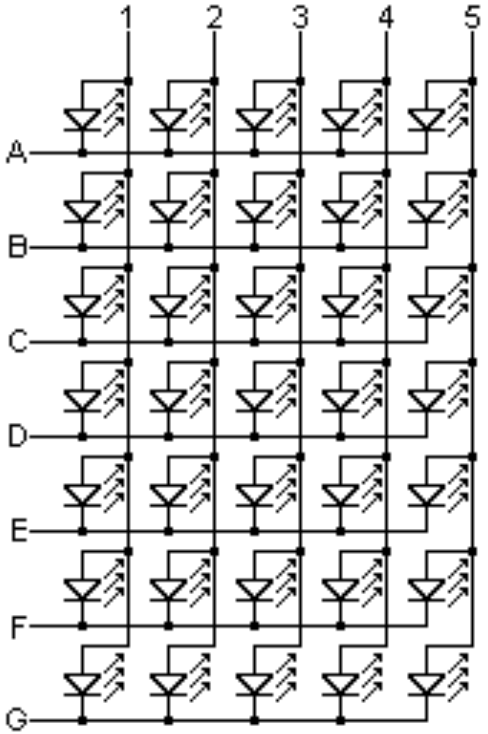
If the FPGA switches faster than 100Hz, it seems that all the displays are lit.



The 8 lower lines control which 7-segment display is enabled.

The lower FPGA lines should go through power transistors (not shown on the drawing).

LED dot-matrix displays



A multiplex driver is required.

For example, you might want to scan the 7 rows, and send data through the 5 columns.

Do that fast enough (100Hz at least) and nobody will notice the multiplexing.

Power transistor might be required (at least for the rows) since FPGA IOs are unlikely to provide enough current.

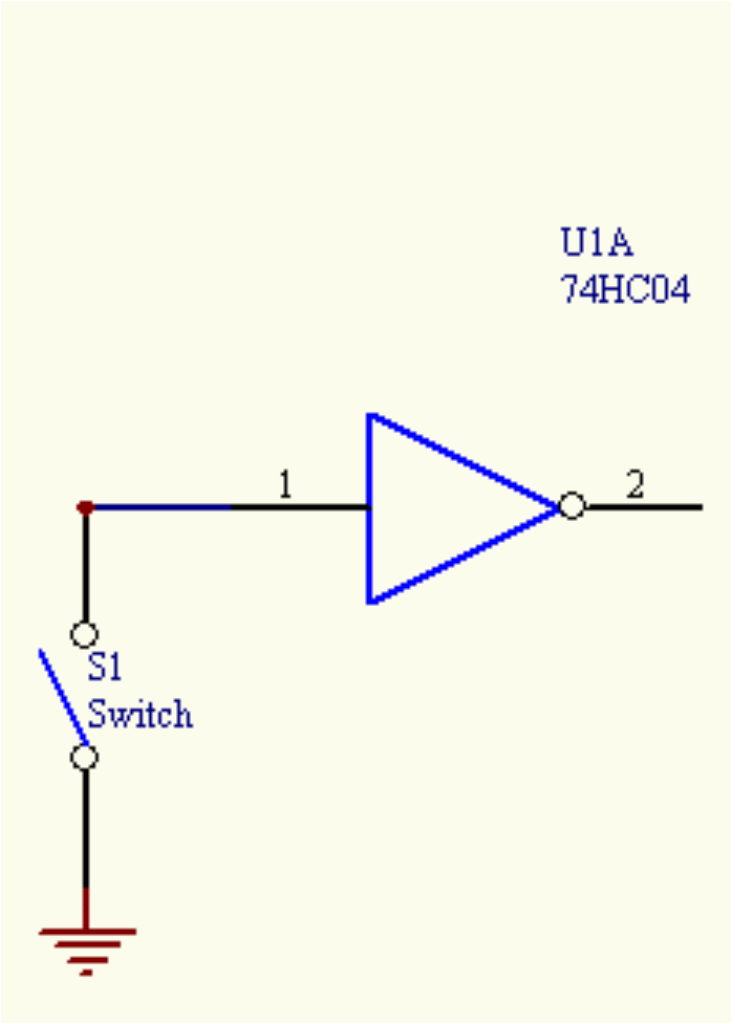
Other devices demanding large current should also be connected through transistors.

The pull-up and pull-down resistors

The basic function of a **pull-up** or a **pull-down** resistor is to insure that given no other input, a circuit assumes a default value.

A pull-up resistor pulls the line high.

A pull-down resistor pulls the line low.



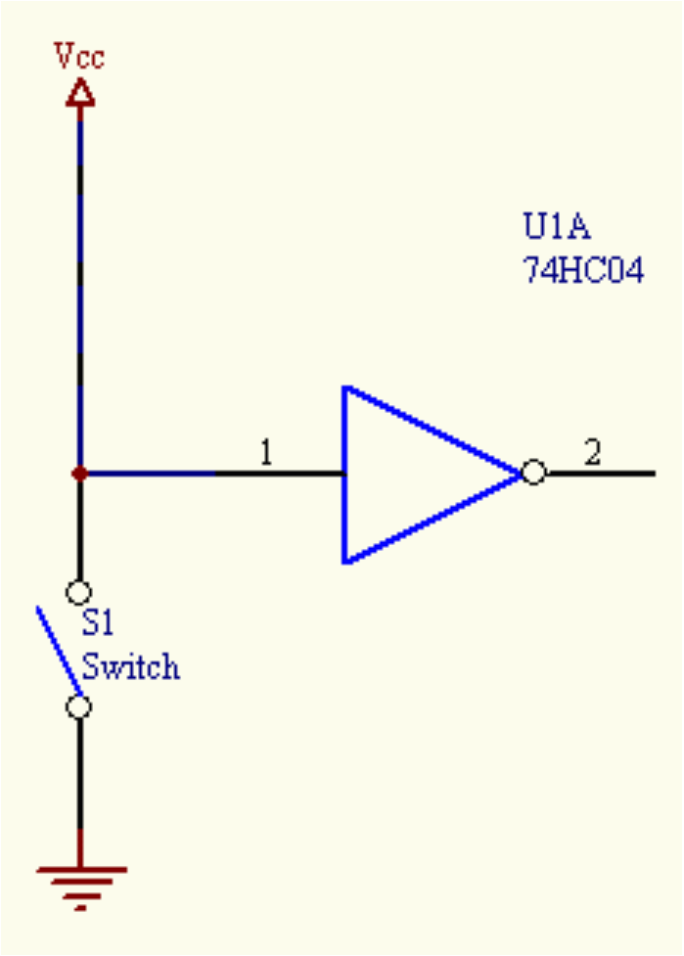
A floating input gate. Not Good!

When switch S1 is closed (on), the input state at pin1 goes low.

When switch S1 is open (off), then input pin 1 is susceptible to a wide array of electrical problems.

The traces or wires connected to pin 1 may allow enough electrical noise in (by acting as little antennas) to cause pin 1 to incorrectly switch states.

What is needed here is a way to connect pin 1 to an electrical potential that can be removed when the switch is closed.



One thought is to tie the pin to Vcc (+5 volts) to insure that pin 1 doesn't float.

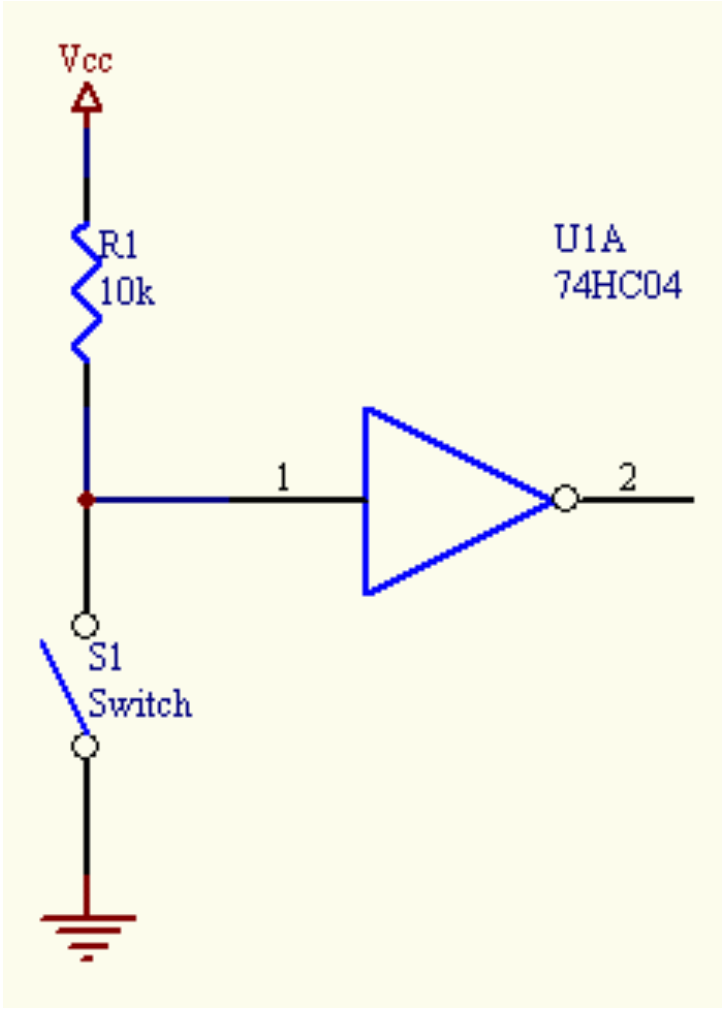
The problem with this circuit is when switch S1 is closed.

This creates a direct electrical connection between Vcc and GND.

Too much current will flow from Vcc to GND.

This causes heat to be generated, which can sometimes burn parts, wires, or even start fires.

In addition, most circuits fail to function correctly because the voltage at the power supply drops to zero.



The same circuit with a pull-up resistor.

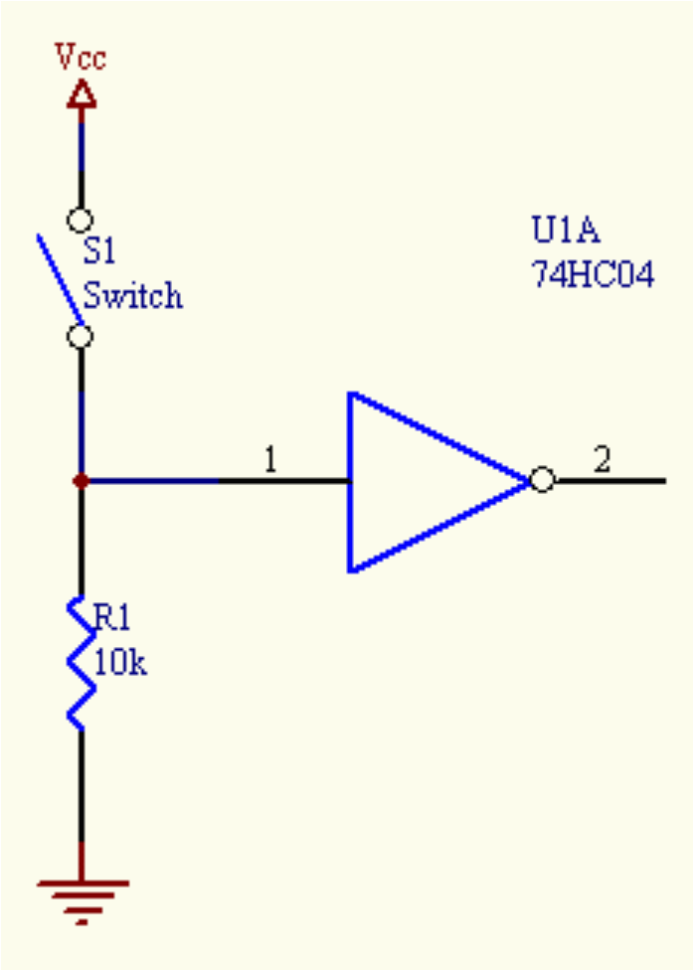
When switch S1 is open (off), pin 1 is tied to Vcc through the resistor.

Since pin1 is a high impedance input, a voltage meter placed on pin 1 will show Vcc (+5v) if connected to pin 1.

When switch S1 is closed (on), pin 1 has a direct connection to GND, which takes it to the low state.

The pin1 side of R1 also has a direct connection to ground.

Current will flow from Vcc, through R1, and to ground but it will be a limited amount of current.



The same circuit with a pull-down resistor.

Just like the pull-up resistor, it is used to limit the current that can flow between Vcc and ground.

Most digital circuits use a 10k or a 47k resistor for pullups.

10k seems to be the most common, but if you are hoping to save as much power as possible, the a 47k resistor may be right for your application.

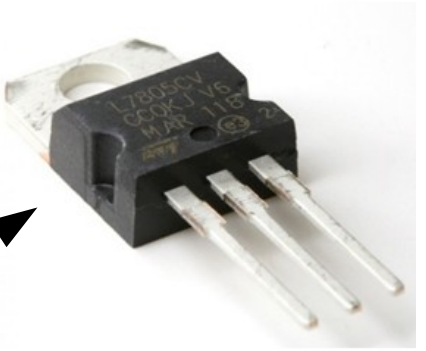
But you don't want the resistance to be too large.

A 4MΩ resistor might work as a pull-up, but its resistance is so large that it may not do its job 100% of the time.

A low resistor value is called a **strong pull-up** (more current flows).

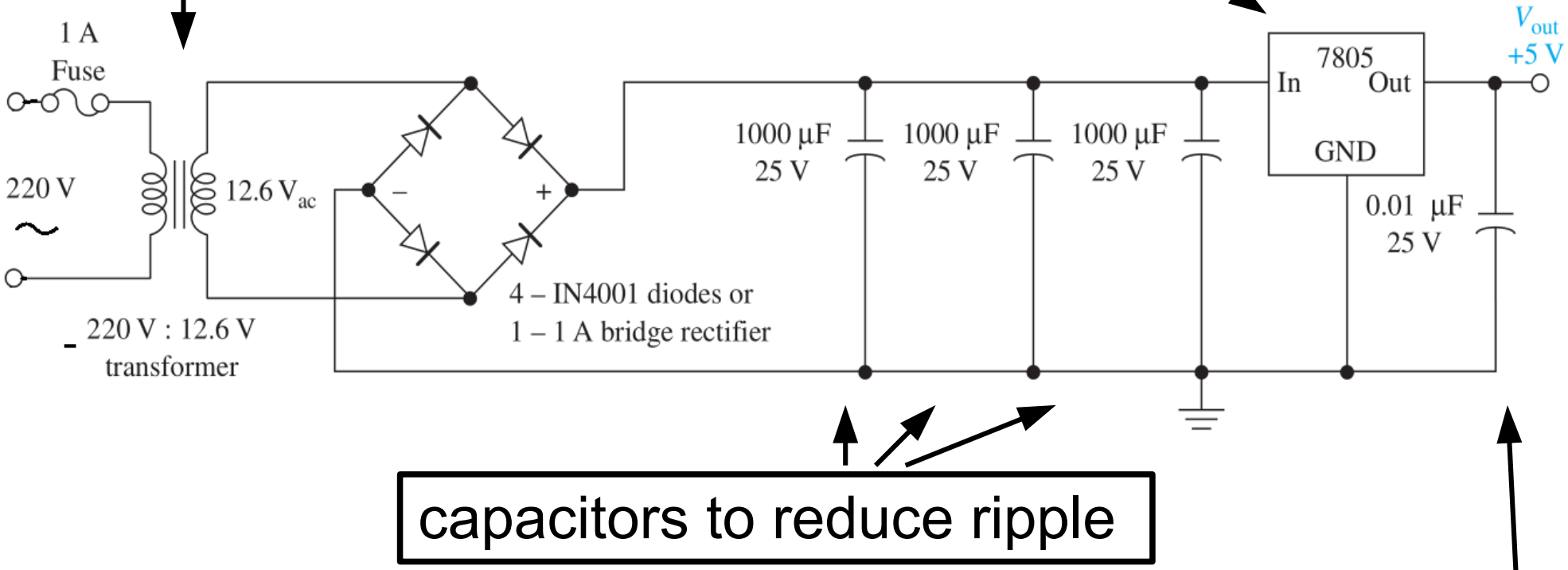
A high resistor value is called a **weak pull-up** (less current flows).

A 5-V power supply



step-down transformer

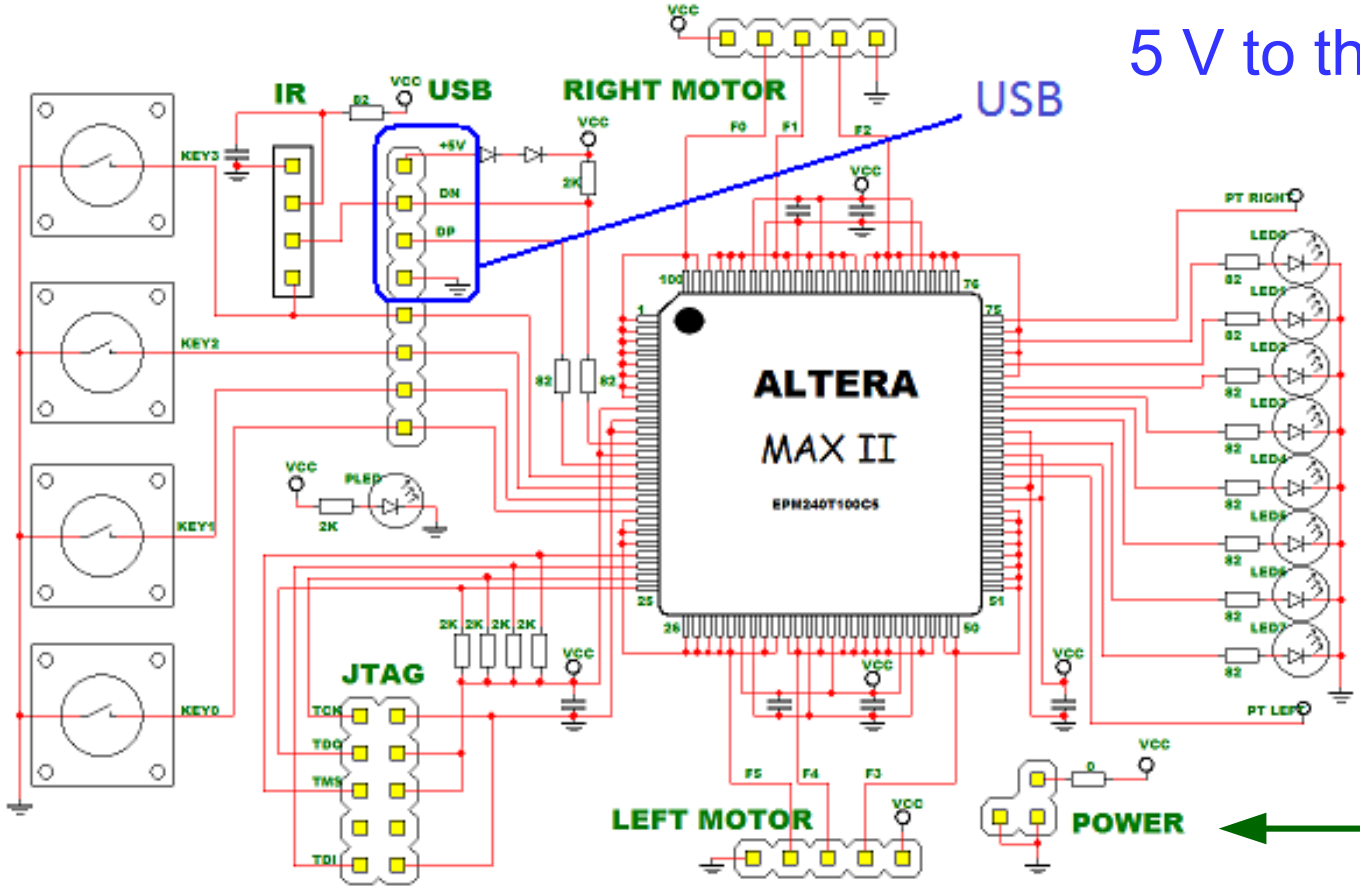
voltage regulator



capacitors to reduce ripple

The decoupling capacitor for eliminating the effects of voltage spikes created from the internal TTL switching and electrostatic noise generated on the power and ground lines.

Power for the Marsohod board



5 V to the USB

from a 5V power supply or from a PC's USB port

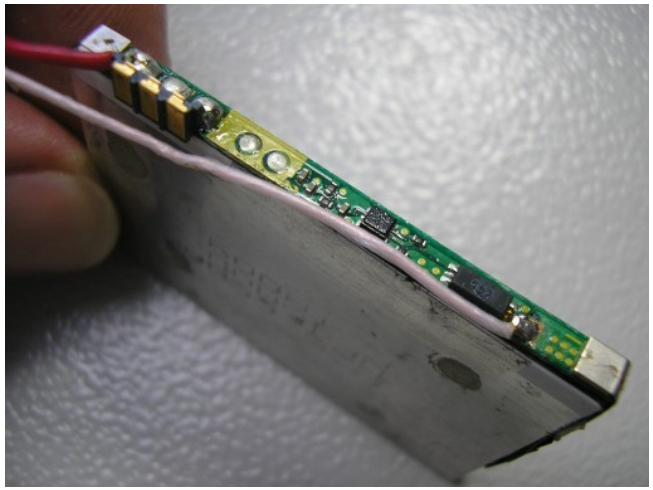
3.3 V here

2.5V to 4.6V

Not more than 4.6V!

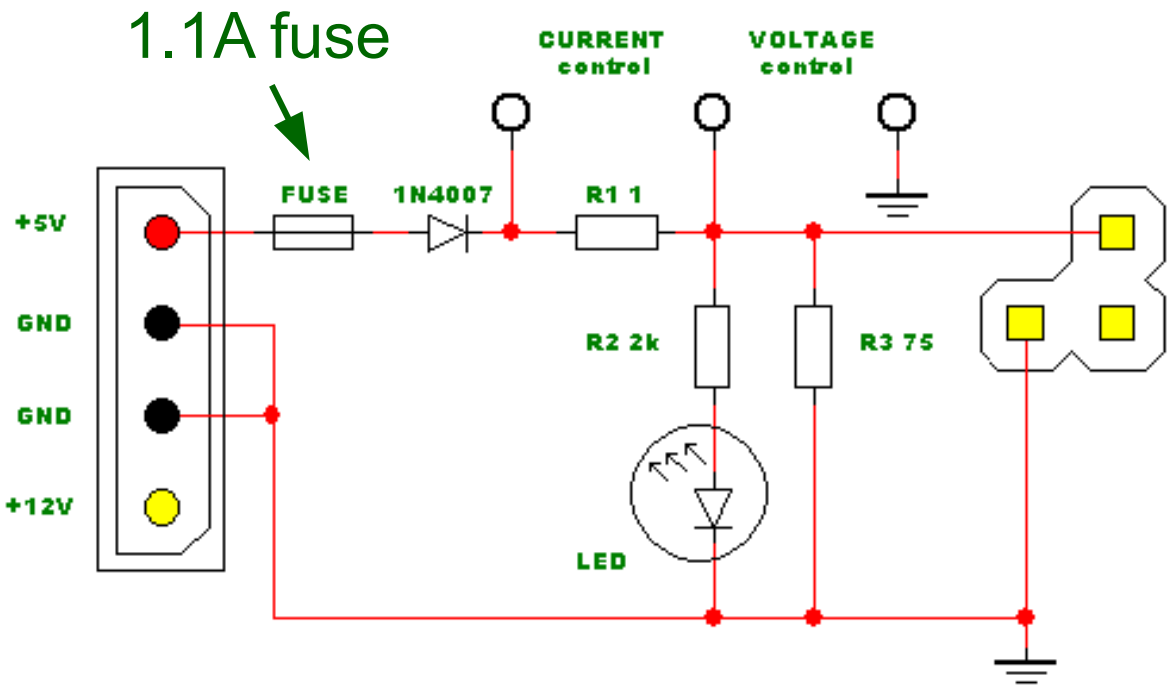
Charging a Lithium-ion battery

from a laptop
or a cell phone



Don't overcharge!

Not more than 4.2V



R3 must be adjusted so that the open output voltage is 4.2V

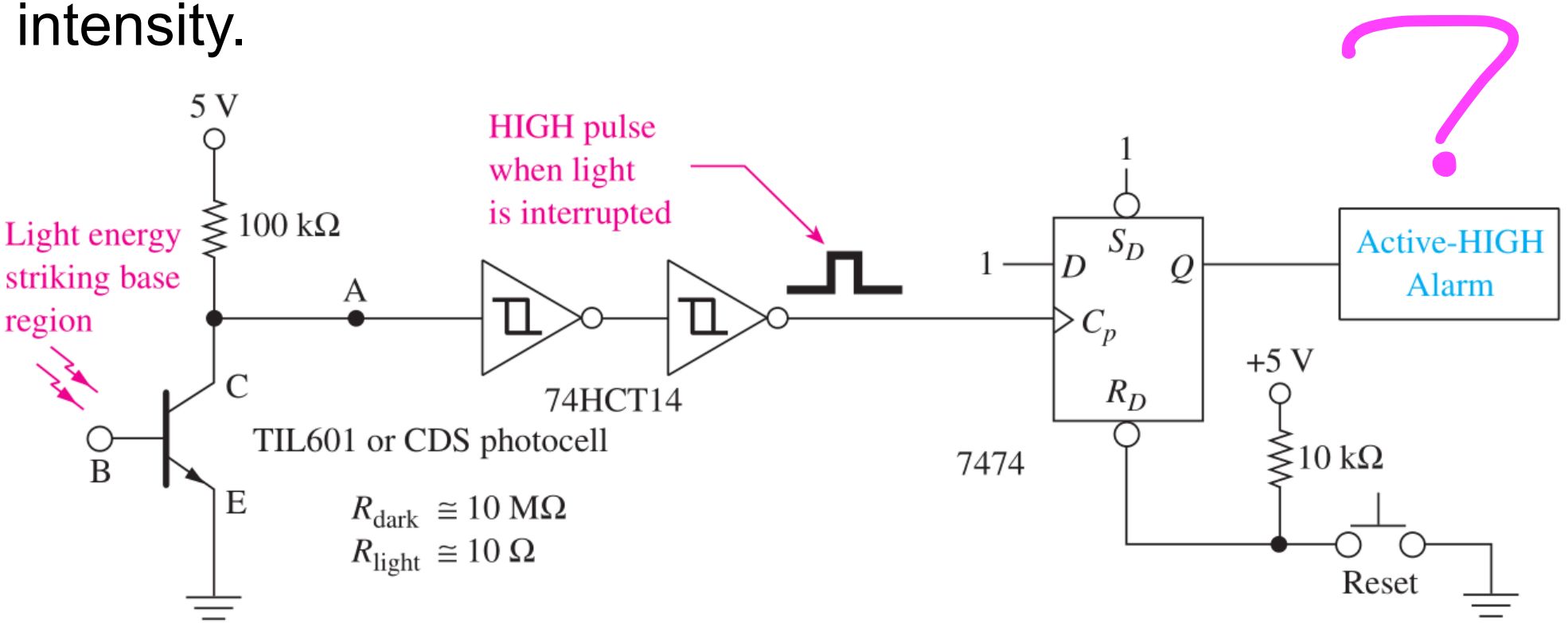
R2 should be a high power resistor or several resistors in parallel, and maybe add a heat sink

Phototransistor input

A phototransistor is made to turn off and on by shining light on its base region.

The resistance from collector to emitter for an OFF transistor is typically 1MΩ to 10MΩ

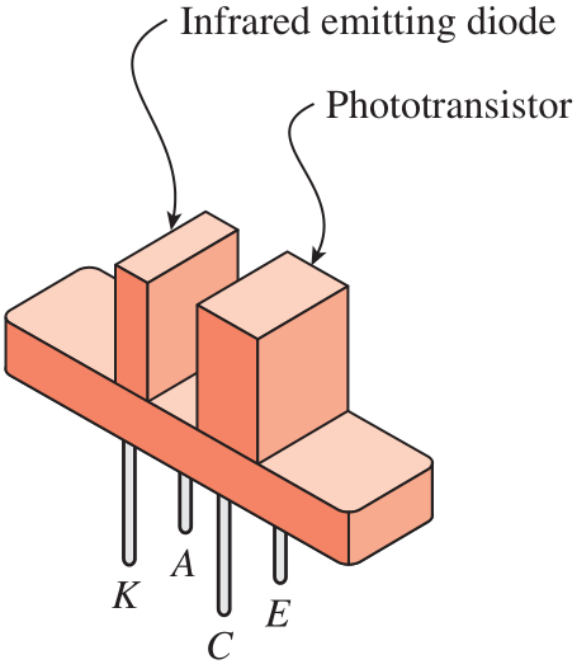
An ON transistor: 1kΩ to 10Ω depending on the light intensity.

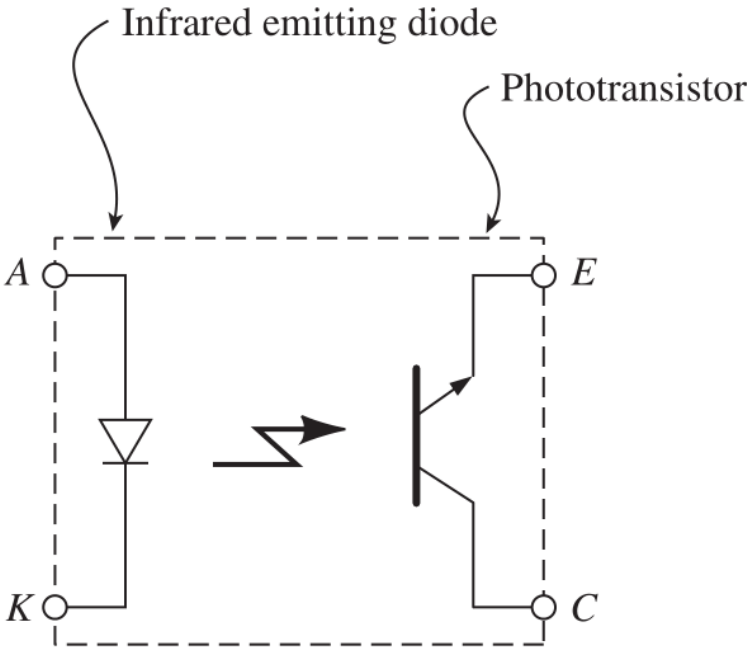


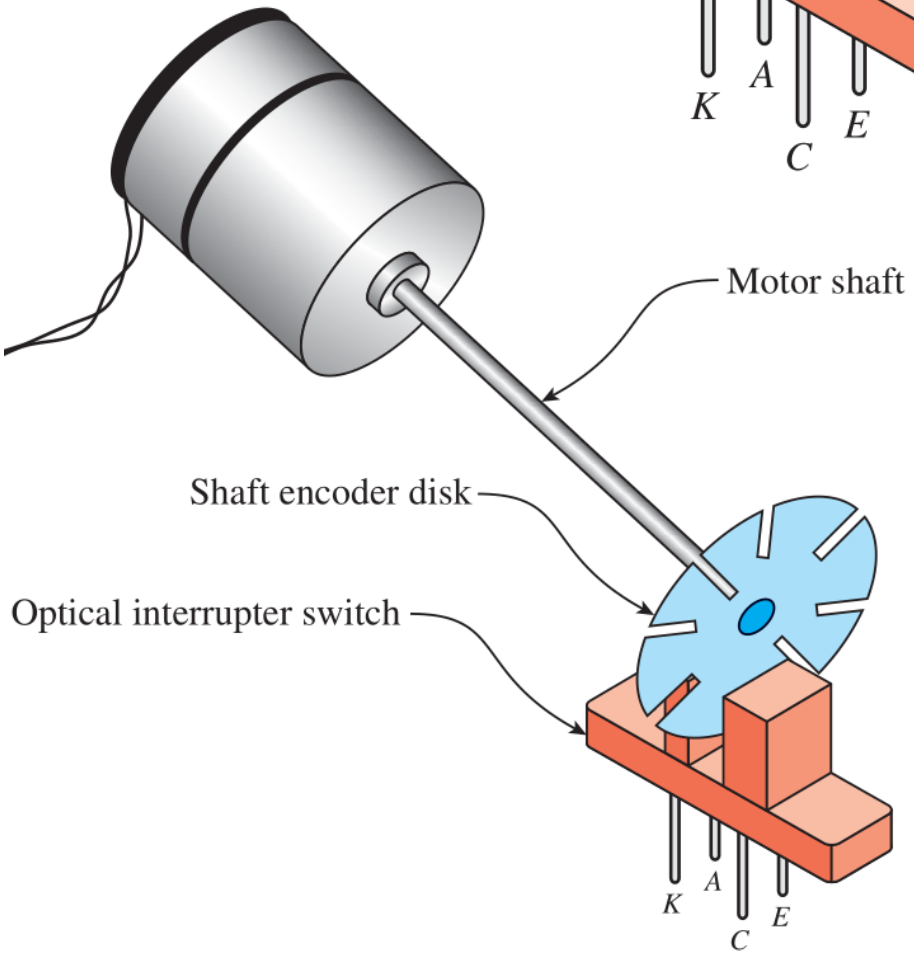
Optocoupler

The output side of the optocoupler is electrically isolated from the input side and can, therefore, be used to couple one circuit to another without being concerned about incompatible or harmful voltage levels.

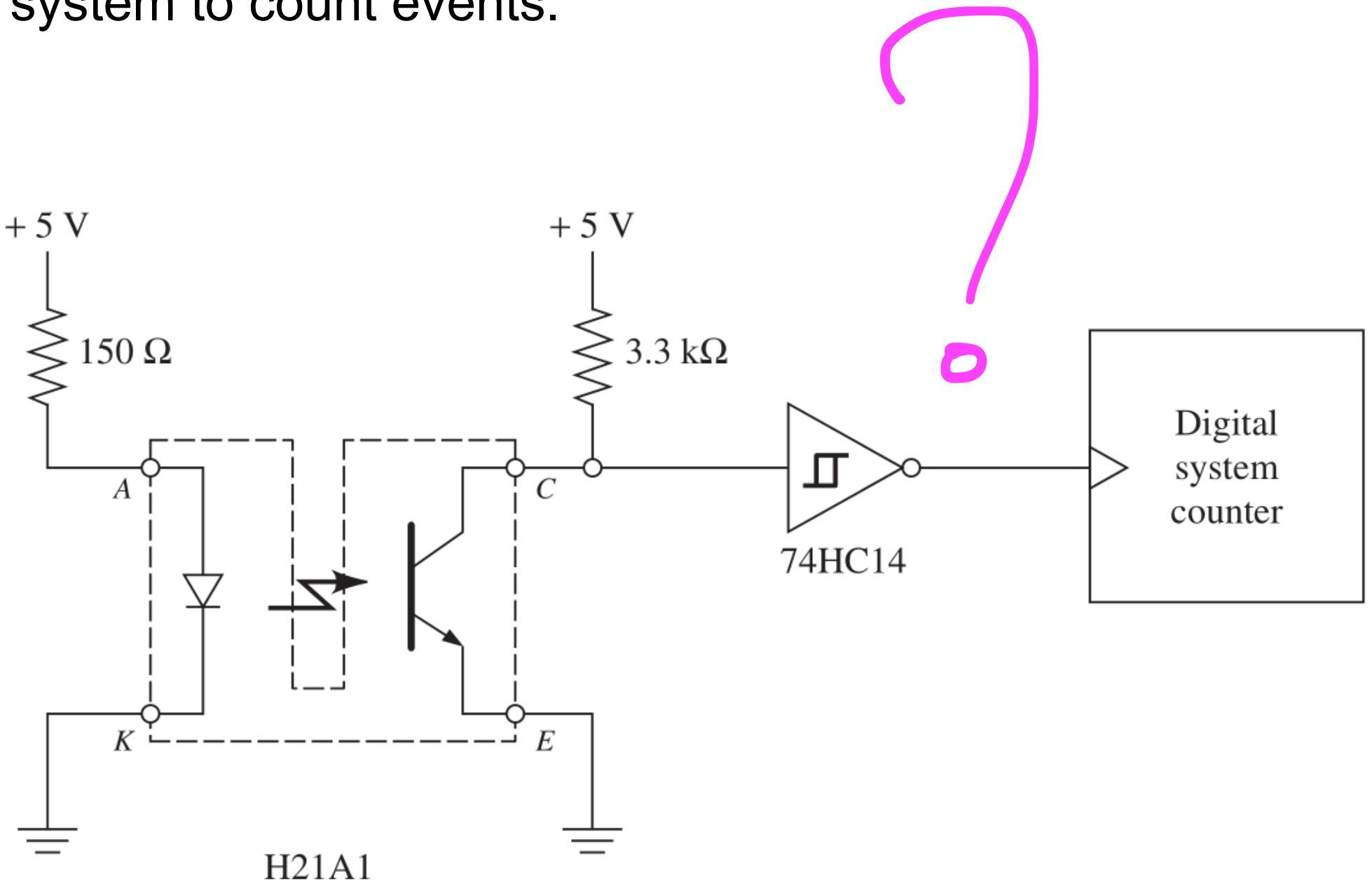
Optical interrupter switch







Connecting the optical interrupter switch in a digital system to count events.



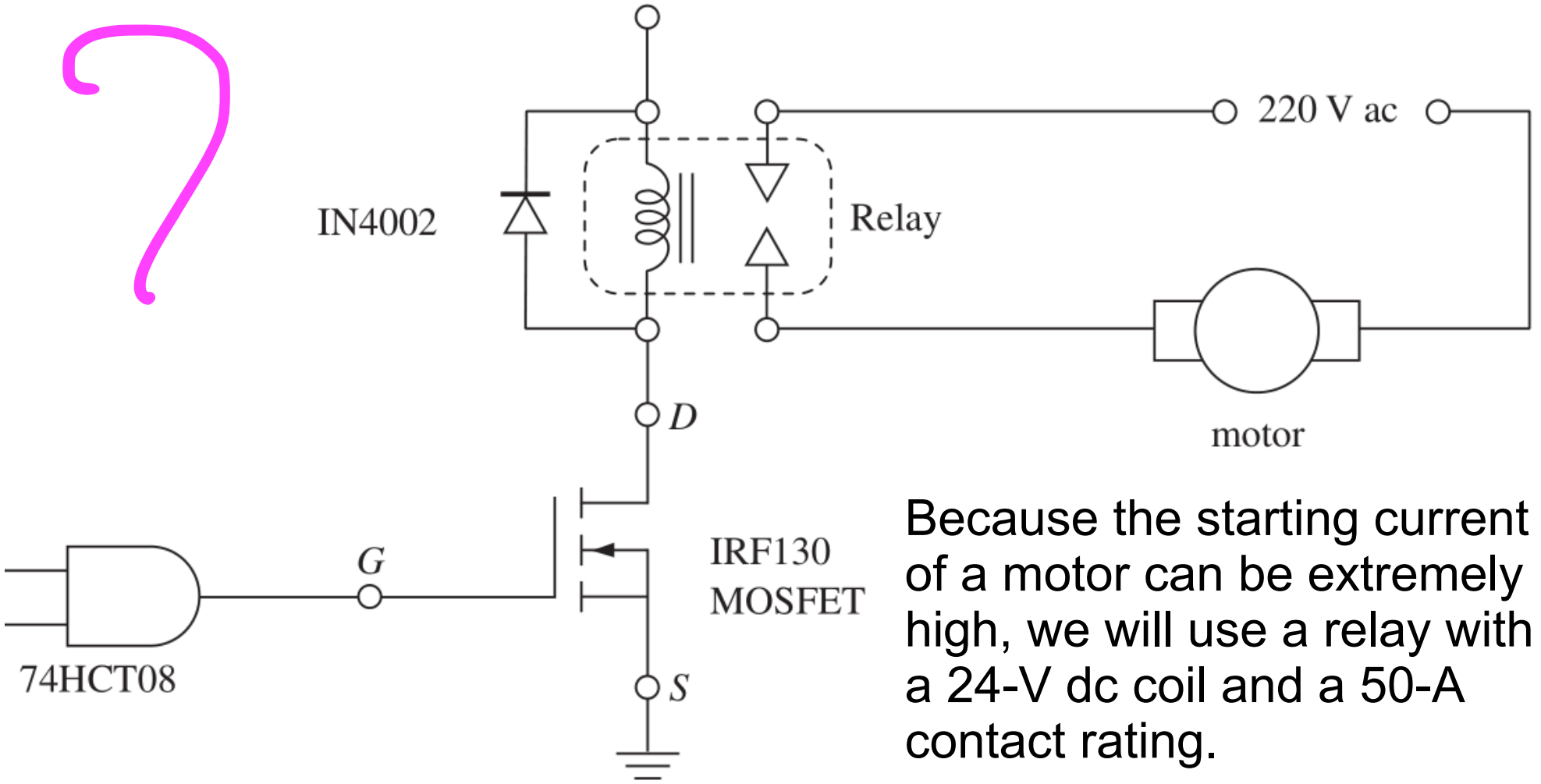
A Power MOSFET used to drive a load

The output drive capability of digital logic severely limits the size of the load that can be connected.

Usually, the output current is 20 – 60 mA depending on the chip.

This is below the current requirements of some loads.

A common way to boost the current capability is to use a power MOSFET – a transistor specifically designed to have a very high input impedance to limit current draw into its gate and also be capable of passing a high current through its drain to source.



Because the starting current of a motor can be extremely high, we will use a relay with a 24-V dc coil and a 50-A contact rating.

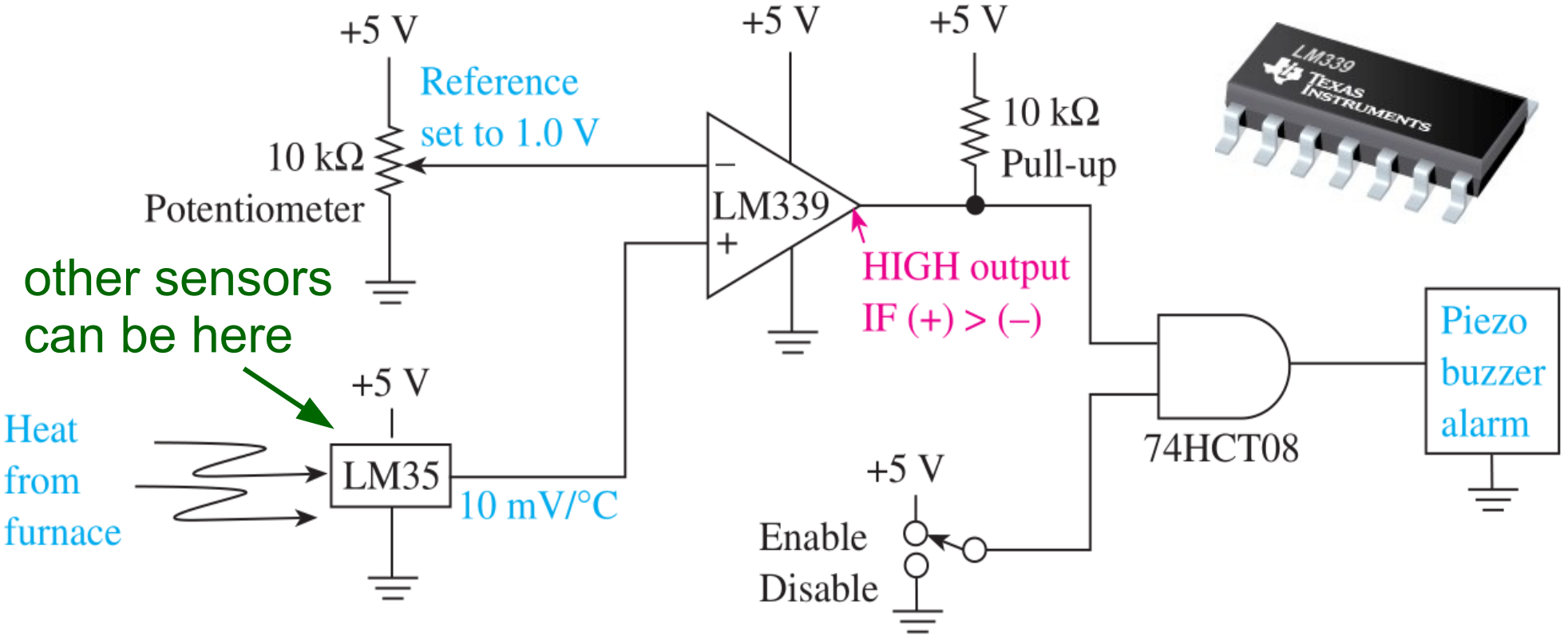
A relay of this size may require as much as 200 mA to energize the coil to pull in the contacts.

A MOSFET such as the IRF130 can pass up to 12 A through its drain to source, so it can handle this relay coil requirement.

Level detecting with an analog comparator

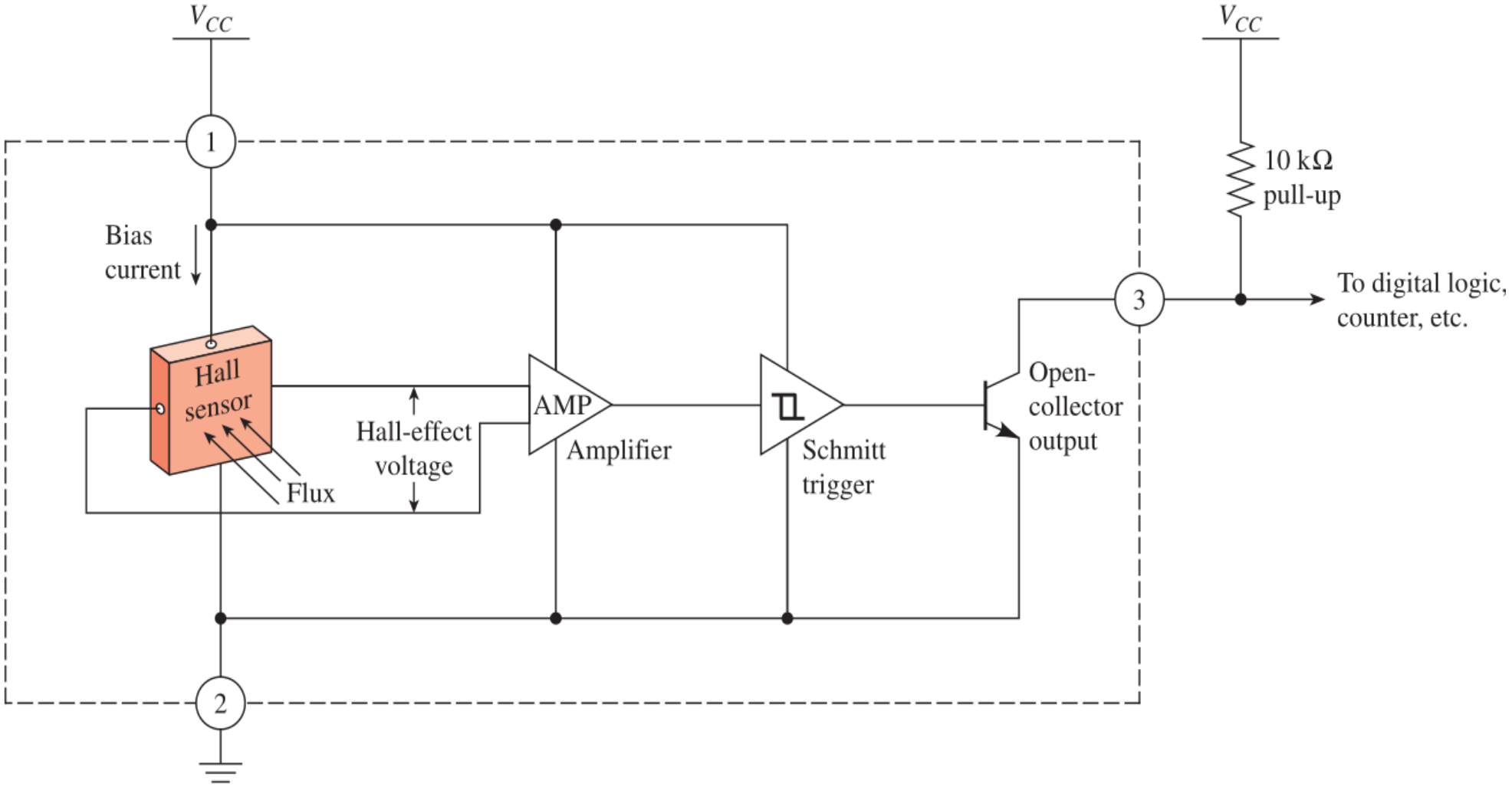
A comparison of the two analog voltages at the comparator's input are used to determine the device's output logic level (1 or 0).

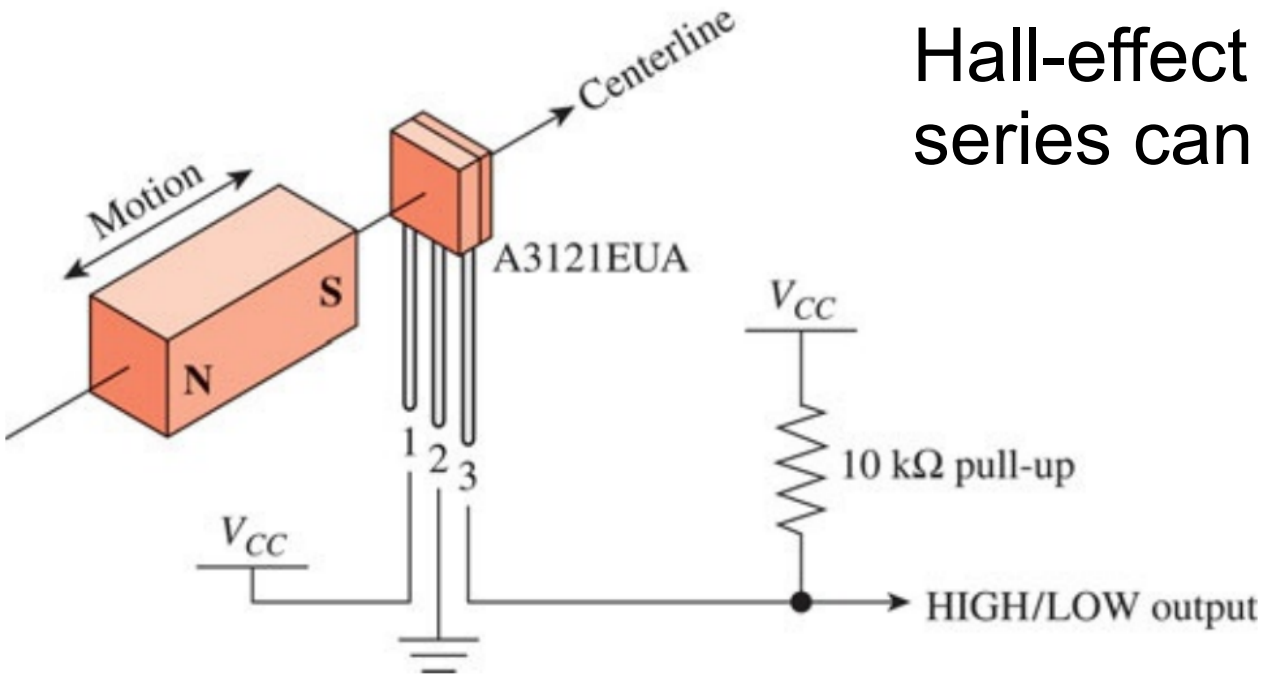
The output of the LM339 acts like an open-collector gate, so a pull-up resistor is required to make the output HIGH.



A Hall-effect switch as a digital input

The Hall effect: a small voltage is output when south magnetic flux lines, perpendicular to the Hall bias current, are brought into close proximity of the Hall sensor (a sheet of semiconductor material).





Hall-effect switches of the 3121 series can run on 4.5 to 24V

When triggered, their open-collector output can sink up to 25 mA.

A pull-up resistor is required to hold the output HIGH when it is not triggered.

A Hall-effect sensor can be used to monitor rotation of a motor.