Synchronous Design

Lecture 04

Josh Brake
Harvey Mudd College

Outline

- Review of basic synchronous design
- Review of the dynamic discipline
- FSM Design Steps
- FSM Activity: Level to pulse converter
- Diode and transistor review

Learning Objectives

By the end of this lecture you should be able to…

- Recall the dynamic discipline and timing specs for designing synchronous digital systems.
- Properly condition asynchronous signals using synchronizers.
- Recall how to use transistors to drive large currents.

Synchronous Digital Systems

- Timing problems are usually the #1 source of difficult bugs
- We can almost completely eliminate the timing problems with a synchronous discipline
  - Like digital vs. analog: digital is a subset of analog
  - Synchronous is a subset of asynchronous timing methodologies
  - Limiting choice makes design easier to understand and avoid sneaky bugs
- Also, will make testability easier (we’ll see that later)

Basic Synchronous Design Rules

- Use only one ______________ (named something clear like `clk`)
- Use only ___________ as state elements (no latches!)
- Put this clock signal into the clock terminal of every flip-flop in the system.

Some common gotchas

Q: How do we begin in a known state?

A: ____________________

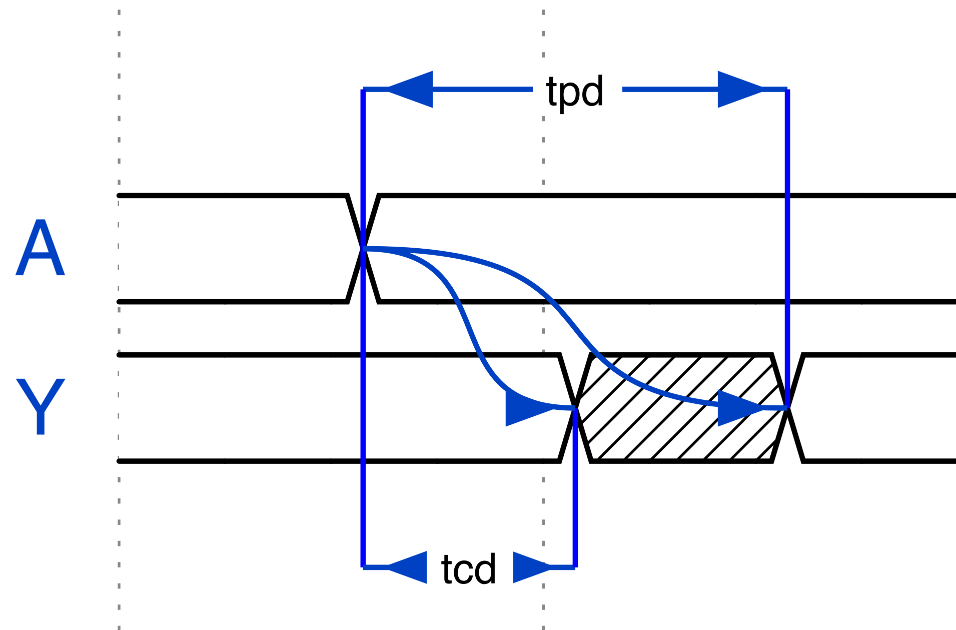
Q: How do we avoid changing the contents of a flip-flop on every clock cycle?

A: _________________

Dynamic Discipline Review

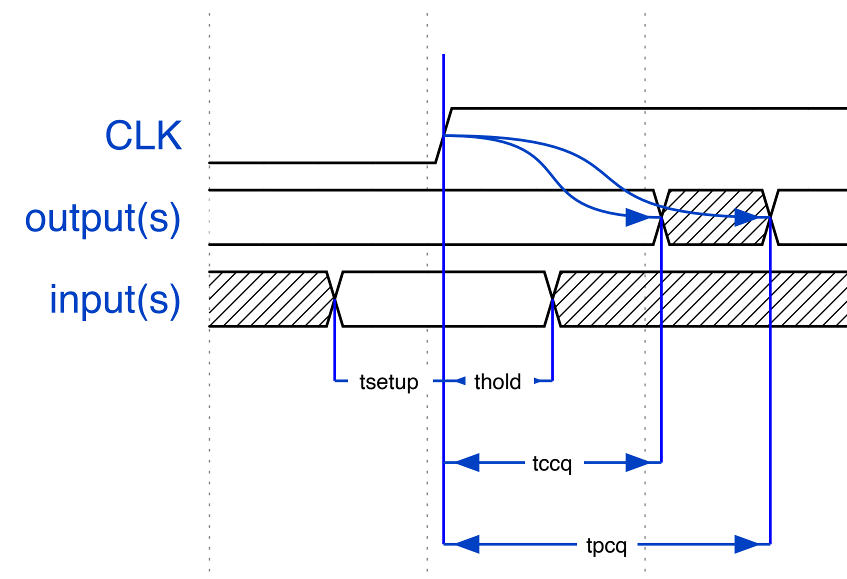
Propagation delay (t_{pd}) – the ____________ time from when an input changes until the output(s) reach their final value.

Contamination delay (t_{cd}) – the ____________ time from when an input changes until any output starts to change its value.



Dynamic Discipline Review: Sequential Logic

- **Propagation Clock-to-Q** (t_{pcq}) – ________ bound on the time from the rising edge of the clock until the output changes.
- **Contamination Clock-to-Q** (t_{ccq}) – ________ bound on the time from the rising edge of the clock until the output changes.
- **Setup time** (t_{setup}) – the amount of time an input to a flop must be stable ________ the clock edge.
- **Hold time** (t_{hold}) – the amount of time an input to a flop must be stable _______ the clock edge.



Synchronous Timing Constraints

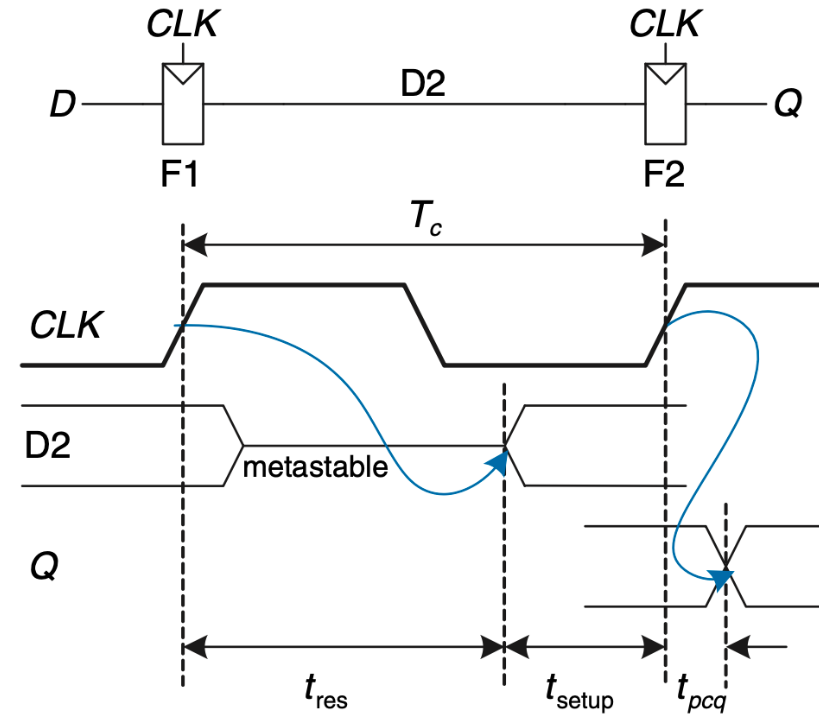
Setup Time Constraint

Hold Time Constraint

Synchronizers

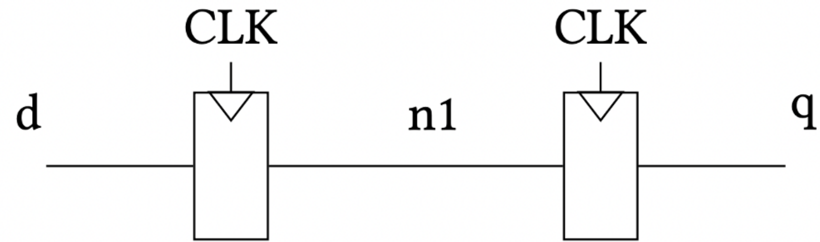
Synchronizer

- The world is asynchronous – how can we cope? Synchronizers!
- Simplest case is a 2-stage synchronizer made of two flops in series.
- If the output of flop F1 goes metastable, we have some time for it to resolve before the next clock edge and the second flop F2.
- This avoids passing metastable inputs out to combinational logic.



Sychronizer

```
1  module sync(input   logic clk,
2              input   logic d,
3              output  logic q);
4
5      logic n1;
6
7      always_ff @(posedge clk)
8      begin
9        n1 <= d;
10       q <= n1;
11     end
12 endmodule
```



Another Sychronizer

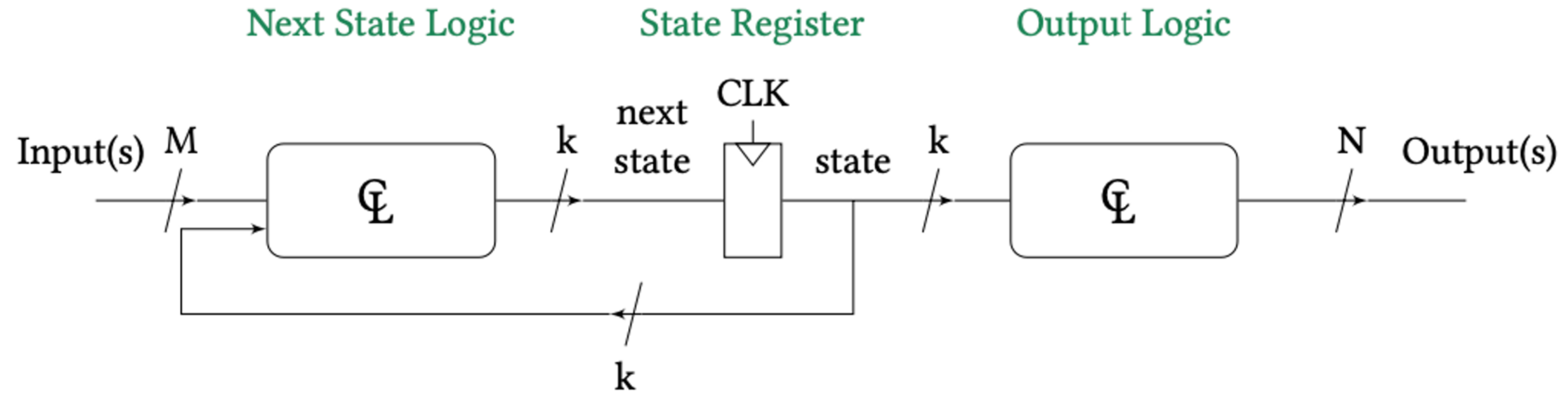
What if I replace the non-blocking assignments with blocking assignments?

What logic does this imply?

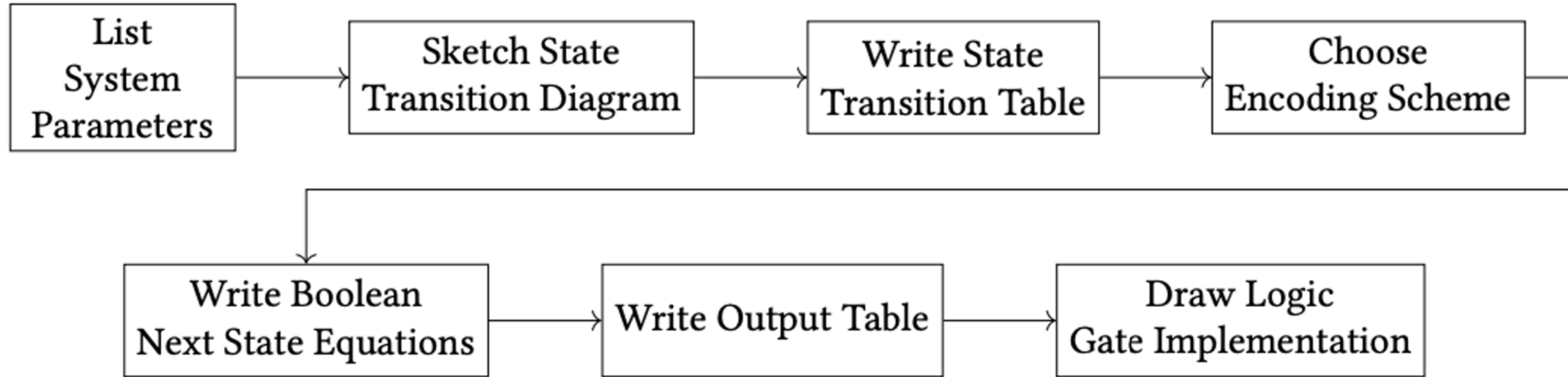
```
 1  module sync(input    logic clk,
 2              input    logic d,
 3              output   logic q);
 4
 5      logic n1;
 6
 7      always_ff @(posedge clk)
 8      begin
 9        n1 = d;
10        q = n1;
11      end
12  endmodule
```

Finite State Machine (FSM) Review

FSM Design



FSM Design Process



FSM Activity

FSM Project: Strobe Signal Generator (Level-to-pulse convertor)

You have been tasked with creating circuitry for a single photon detector. When a photon arrives, it generates a pulse of a random length. We want to generate an output pulse of a fixed duration whenever a photon hits the detector.

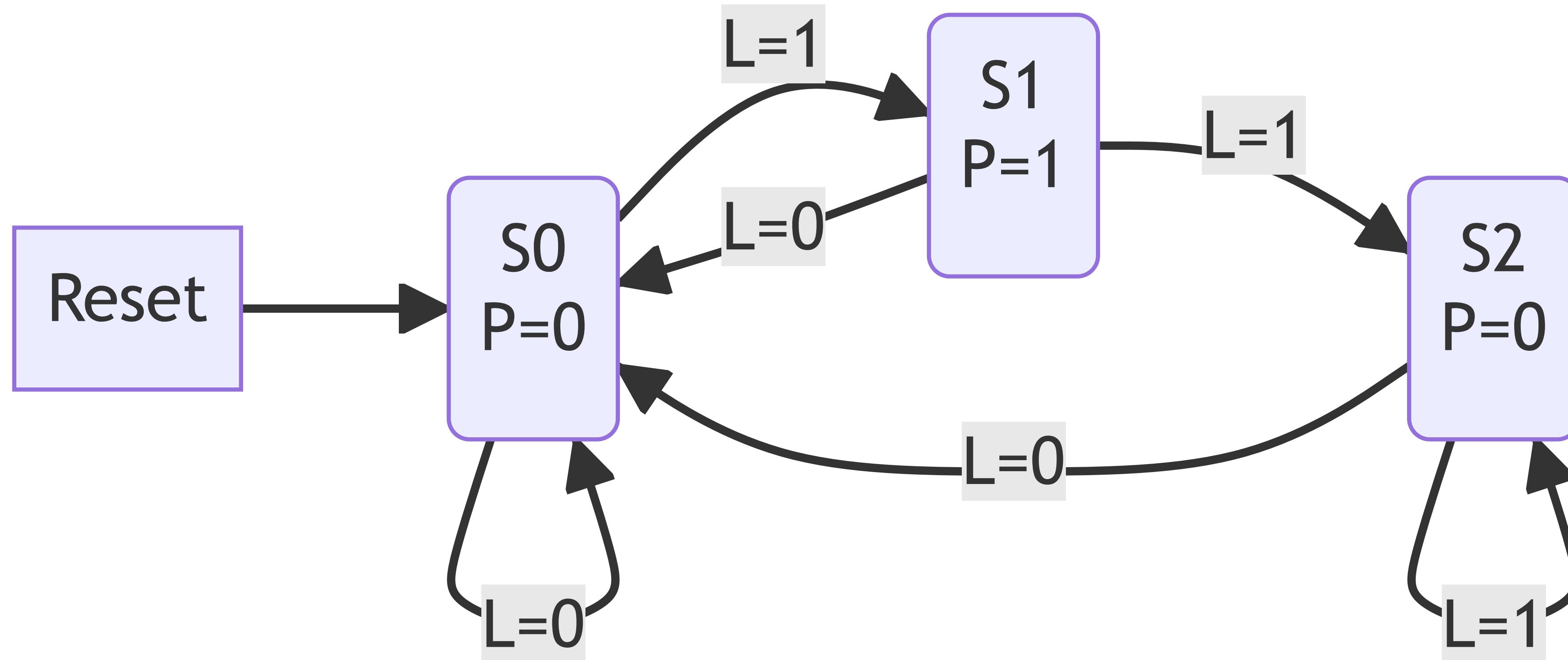
Your task (should you choose to accept it): Design an FSM which generates a pulse for a single clock cycle when an output goes from low to high. Add a synchronizer to ensure that the input does not cause metastability.

List Out Specifications

Inputs

Outputs

State Transition Diagram



State Transition Table

Output Logic

FSM Verilog Template

Five elements:

1. Inputs and outputs
2. Internal signal definition
3. State register: `always_ff` block. Make sure you have a reset!
4. Next state logic: `always_comb` block or `assign` statements.
5. Output logic: `always_comb` block or `assign` statements

FSM Verilog Template

Module and signal declaration.

```
1  // This module converts a level change on an input signal to
2  // a single clock cycle output pulse.
3
4  module level_to_pulse_converter(
5      input  logic clk, reset,
6      input  logic L,
7      output logic P
8  );
9
10     logic [2:0] state, nextstate;
11
12     parameter S0 = 3'b001;
13     parameter S1 = 3'b010;
14     parameter S2 = 3'b100;
15
16     // Could also use something like the following for specifying the
17     // state encodings.
18     //   typedef enum logic [1:0] {S0, S1, S2} statetype;
19     //   statetype state, nextstate;
20  ...
```

FSM Verilog Template

State register.

```
...
   // State register
  always_ff @(posedge clk, posedge reset)
      if (reset)  state <= S0;
      else        state <= nextstate;

...
```

FSM Verilog Template

Next state and output logic.

```
...
    // Next state logic
    always_comb
        case (state)
            S0: if(L) nextstate = S1;
                else  nextstate = S0;
            S1: if(L) nextstate = S2;
                else  nextstate = S0;
            S2: if(L) nextstate = S2;
                else  nextstate = S0;
            default: nextstate = S0;
        endcase

    // Output logic
    assign P = (state == S1);
endmodule
```

Develop a testbench for this project

Steps to create a testbench

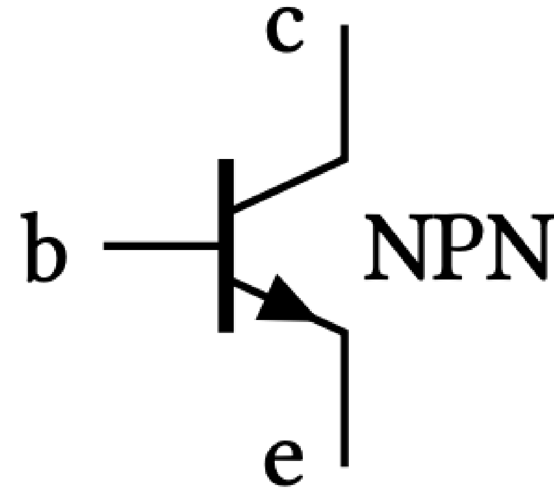
1. Create clock signal which toggles continuously for any synchronous elements.
2. Initial statement to apply reset and set inputs to desired initial values.
3. Another initial block to apply input signals.

Don’t apply signals on a clock edge! (e.g., make sure that if you are using a clock period of 10 timesteps that you don’t apply your inputs at multiple of 10.

Testbench Code

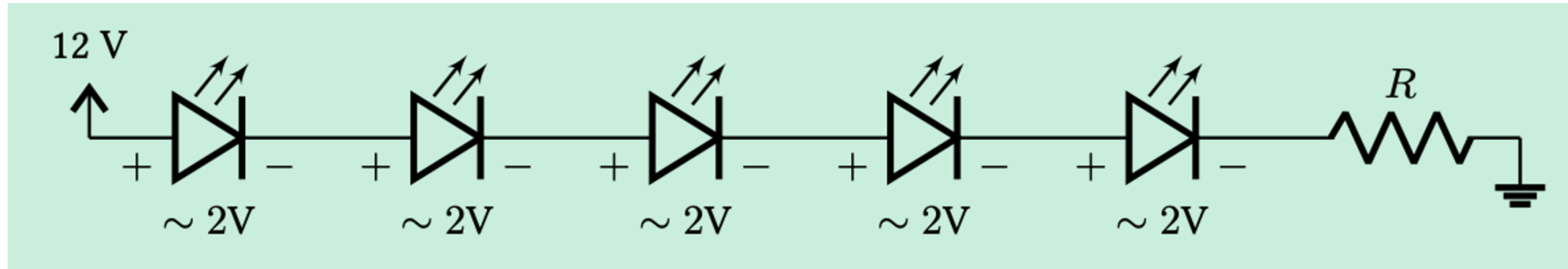
Diode and Transitor Review

- $i_D = i_0 \left(\exp\left(\frac{v_D}{n \cdot v_T}\right) - 1\right)$
- n and i_0 are scaling factors and v_T is the thermal voltage which is $v_T = kT/q$ (25.4 mV at room temperature).
- For a silicon diode, $v_{on} \approx 0.7V$ and for an LED $v_{on} \approx 1.7 - 2.1V$



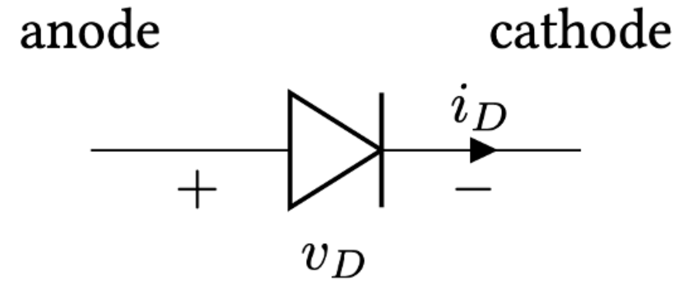
NPN Symbol

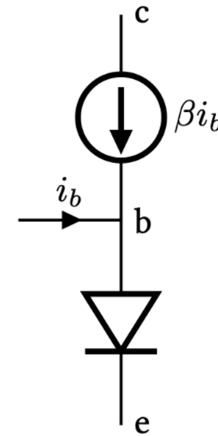
How many LEDs can you light up in series from a 12 V source?



Transistors

Used to pull load ______. (i.e., connect load to _________)



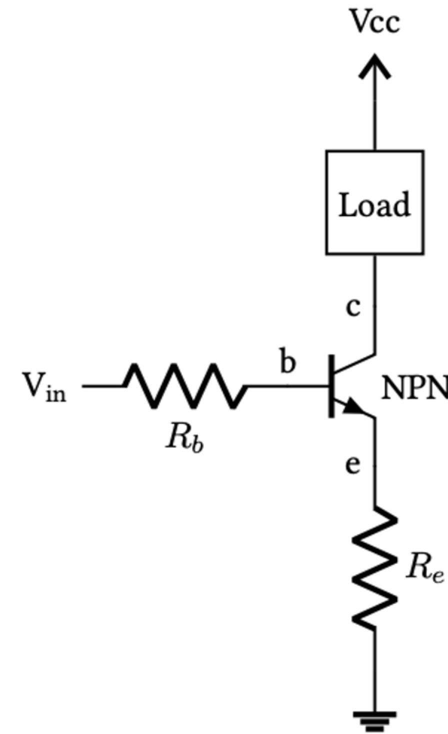


BJT small signal model

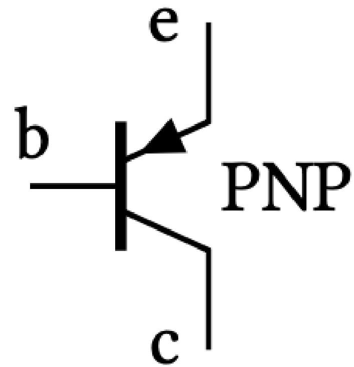
Driving a load with an NPN transitor

How do we choose R_b and R_e?

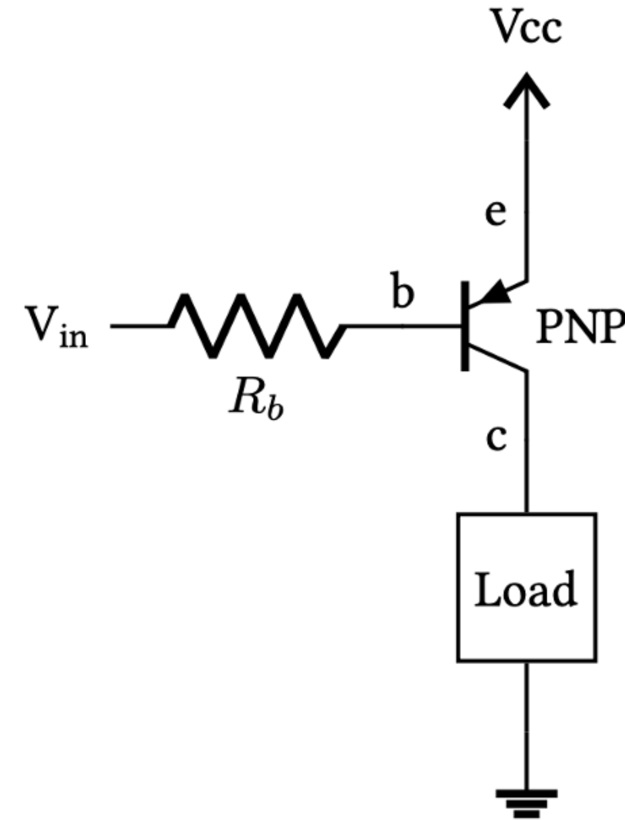
Set R_e to zero.



Driving a load with an PNP transitor



PNP Transistor Symbol



Used to pull load ______. (i.e., connect load to ________)

Wrap up

- Synchronous sequential design enables us to design simple and robust digital systems.
  - Only one clock signal to all flops (single clock domain)
  - Ensure that the setup and hold time constraints are observed.
- We need to synchronize asynchronous inputs to avoid metastability. Price is an additional clock cycle of latency.
- Transistors are like electrically controlled switches and enable us to drive larger loads from weak source (e.g., FPGA/MCU I/O pins)

Announcements/Reminders

- Checkoffs continue today – don’t delay starting on Lab 2. Can reuse code from Lab 1
  - Only **one** seven_seg Verilog module.
  - Make sure LEDs are consistent brightness no matter how many segments are on
  - Develop a testbench to confirm your circuit is working. See tutorial on the website.
- Next week: FPGA documentation and intro to the MCU