

Power II

6.9000

Spring 2025

All Computation Requires Power

- Power is related to energy
- All computation uses energy
- For a given computational technology...
 - The more computation you do, the more energy you use
 - The faster you do your computation, the more energy used per unit time, the more *power* your system uses

Lower Limit on Computation

- There is a lower limit: it takes about 3×10^{-21} Joules to erase a bit no matter what
 - Called Landauer Limit
 - Experimentally shown in 2012 (Berut et al., Nature 2012)
- Intel 22nm process takes approximately
 - 100×10^{-15} Joules (estimate/approximation)
- Between those two numbers are the *inefficiencies* and *limitations* of circuits
- People actively working on pushing towards that limit!...many people at MIT

<https://spectrum.ieee.org/computing/hardware/landauer-limit-demonstrated>

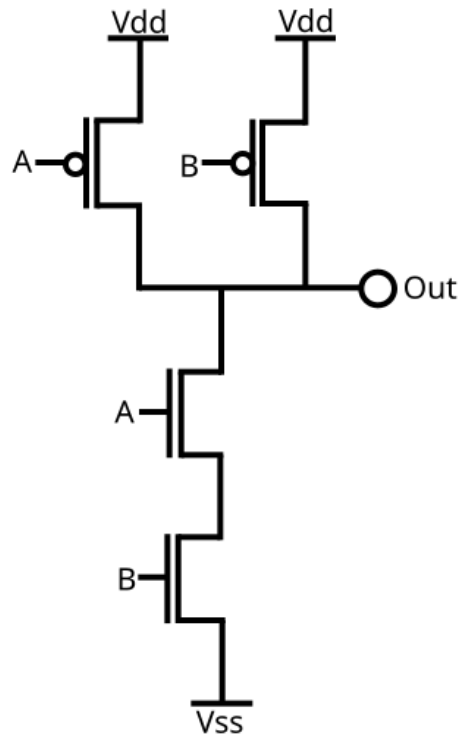
https://en.wikipedia.org/wiki/Landauer%27s_principle

What Are Those Inefficiencies of Modern Digital Circuits?

- *Why* do our modern computing devices use the power that they do?
- Important to remember the famous statement...
“All Models are wrong...some models are useful.”

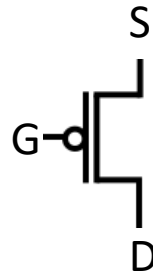
https://en.m.wikipedia.org/wiki/File:CMOS_NAND.svg

Basic Modern CMOS NAND Gate



A	B	Out
0	0	1
0	1	1
1	0	1
0	0	1

P-Channel FET



N-Channel FET

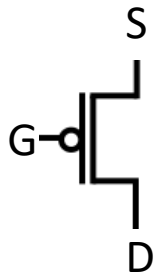


https://en.m.wikipedia.org/wiki/File:CMOS_NAND.svg

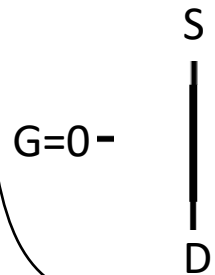
Two Types of Transistors

- Both Types of transistors can act as voltage-controlled-switches.
- Two Transistors work in opposite...dare I say... “complementary” ways

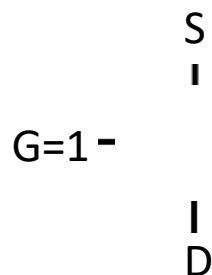
P-Channel FET



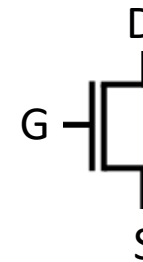
When $V_G = 0$,
device conducts



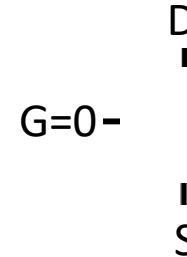
When $V_G = 1$,
device open



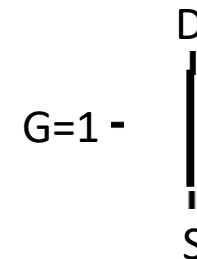
N-Channel FET



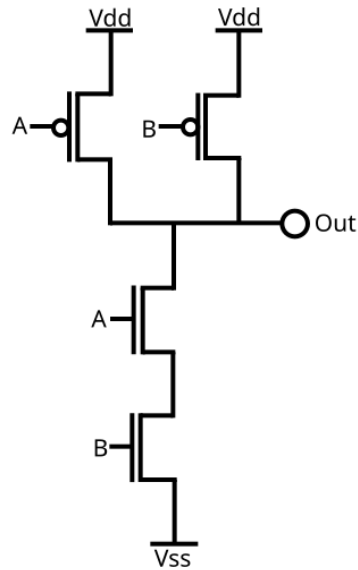
When $V_G = 0$,
device open



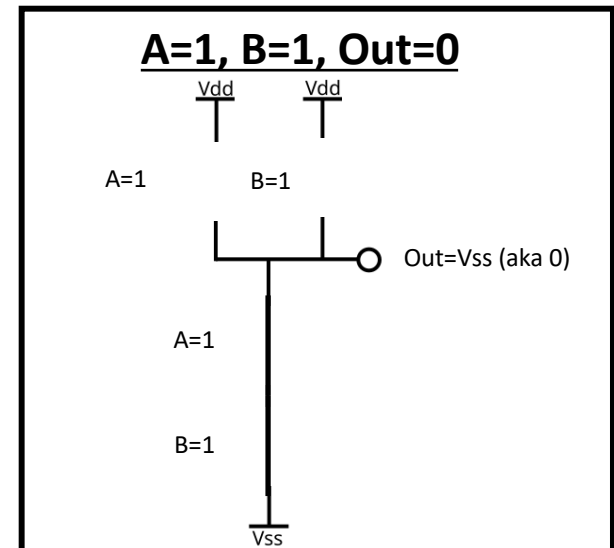
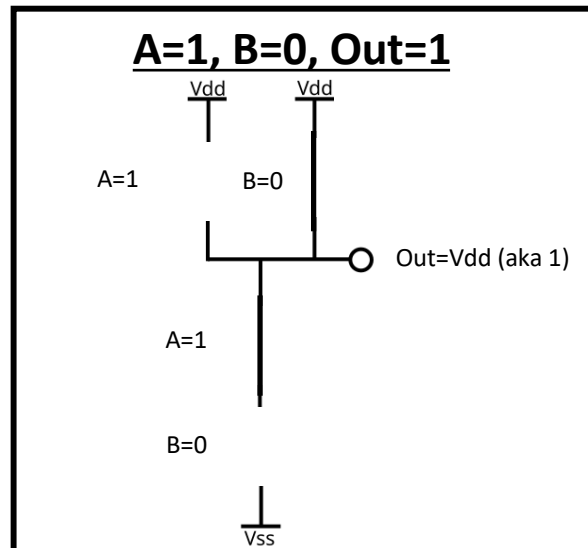
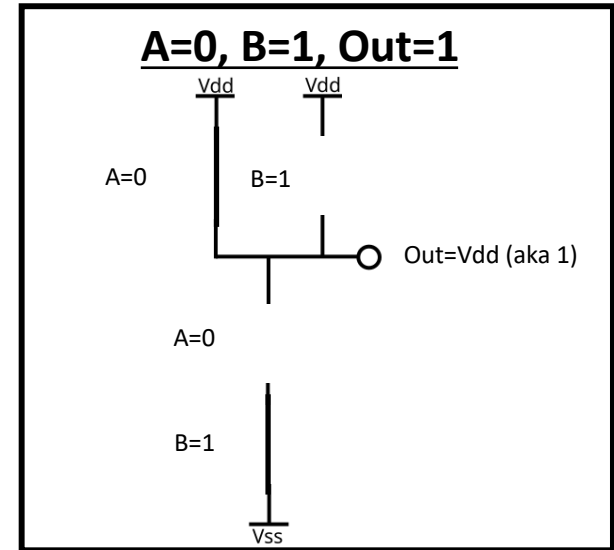
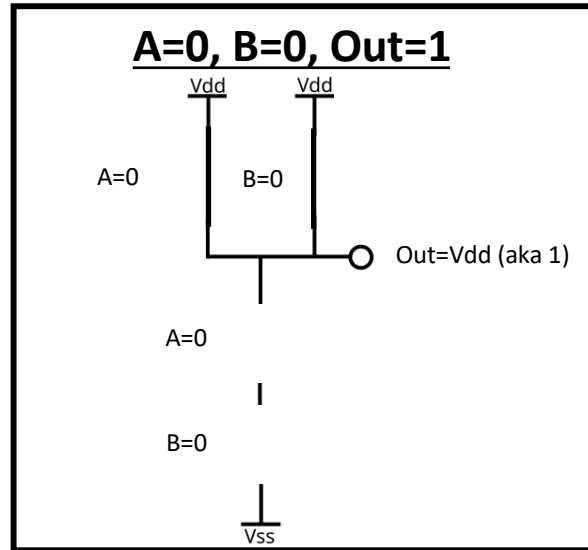
When $V_G = 1$,
device conducts



Theoretical NAND Gate Operation

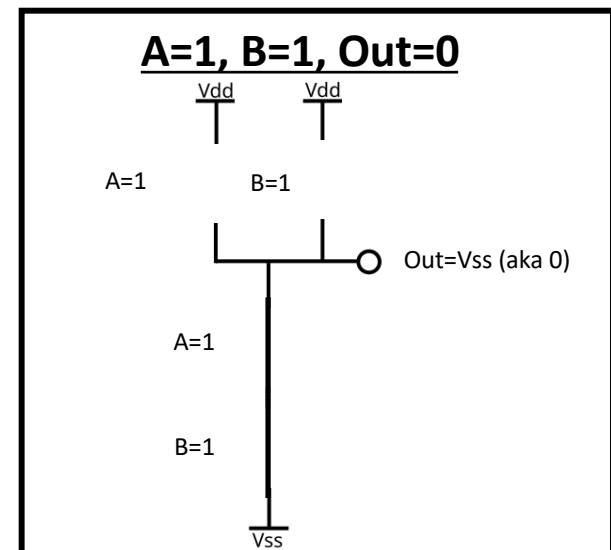
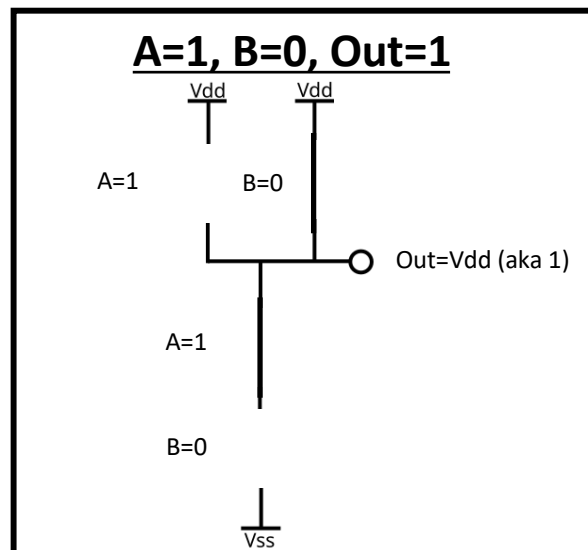
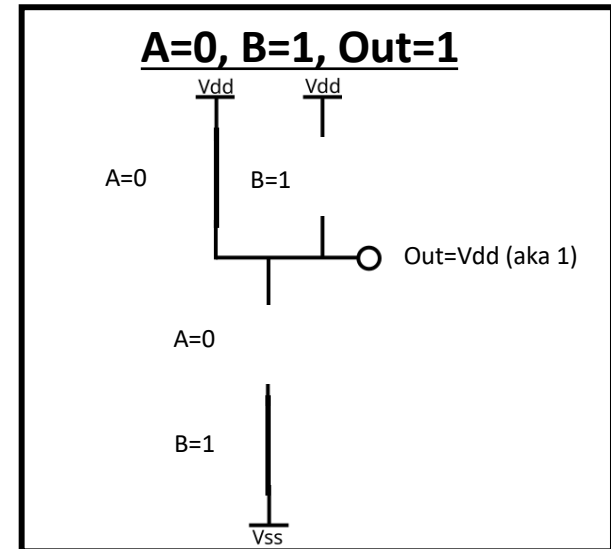
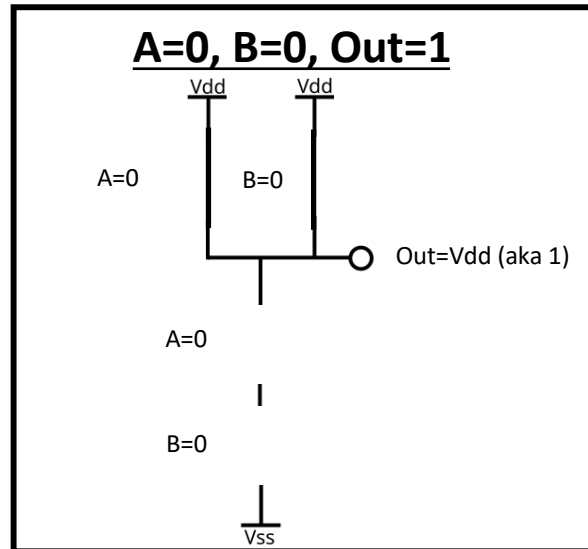


A	B	Out
0	0	1
0	1	1
1	0	1
0	0	1



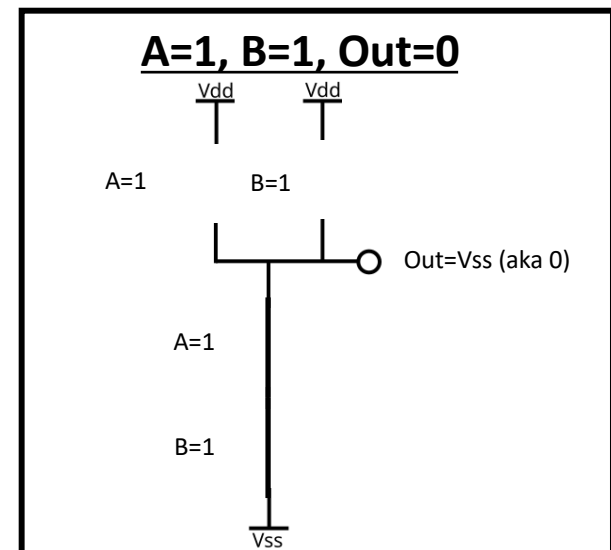
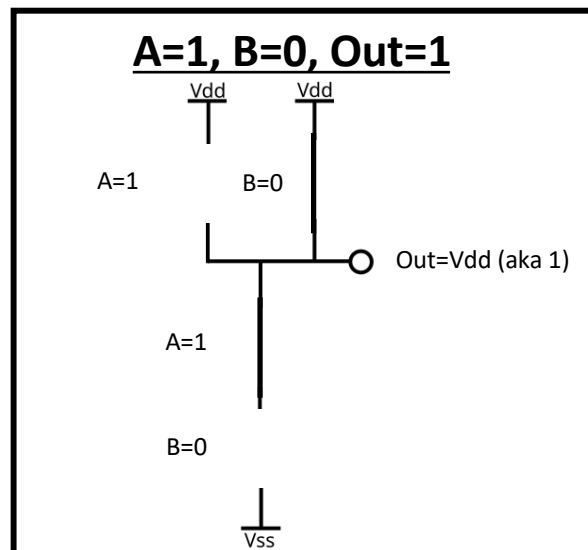
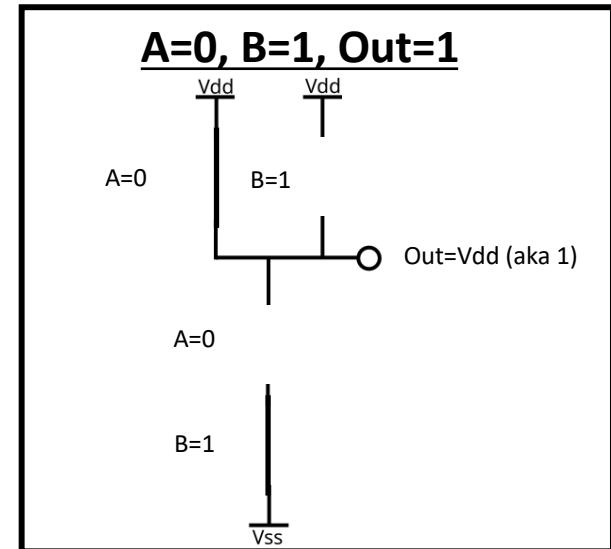
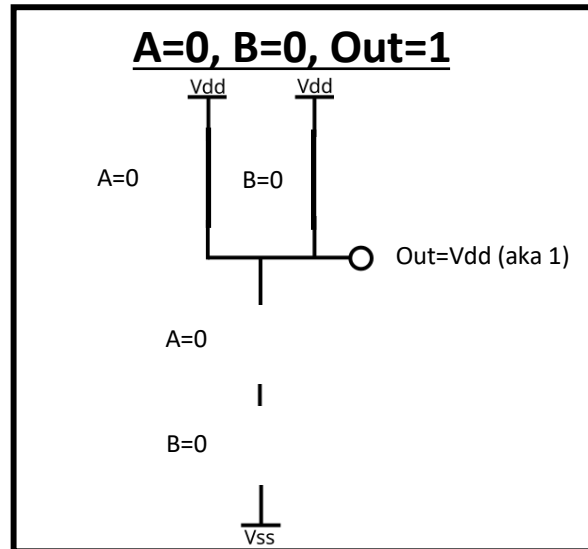
Characteristics...

- In None of these four circuit states is there a direct path from Vdd to Vss
- So one should expect no current flow in the circuit itself
- Therefore no **static power consumption**



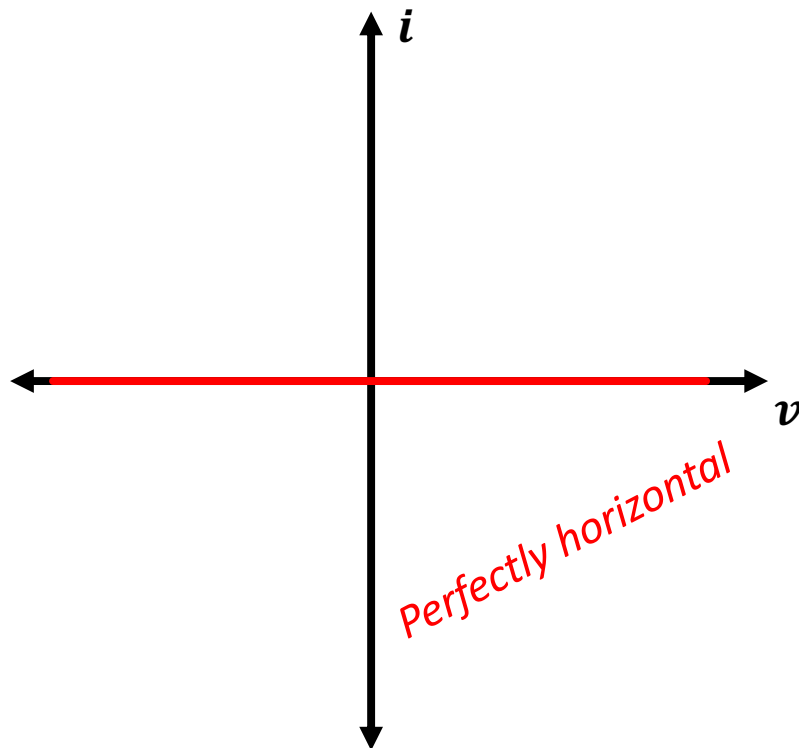
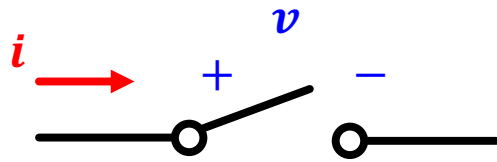
Characteristics...

- It takes no energy to turn on/turn off a switch... They're “massless”
- We can transition between states instantaneously
- Therefore no *dynamic power consumption*

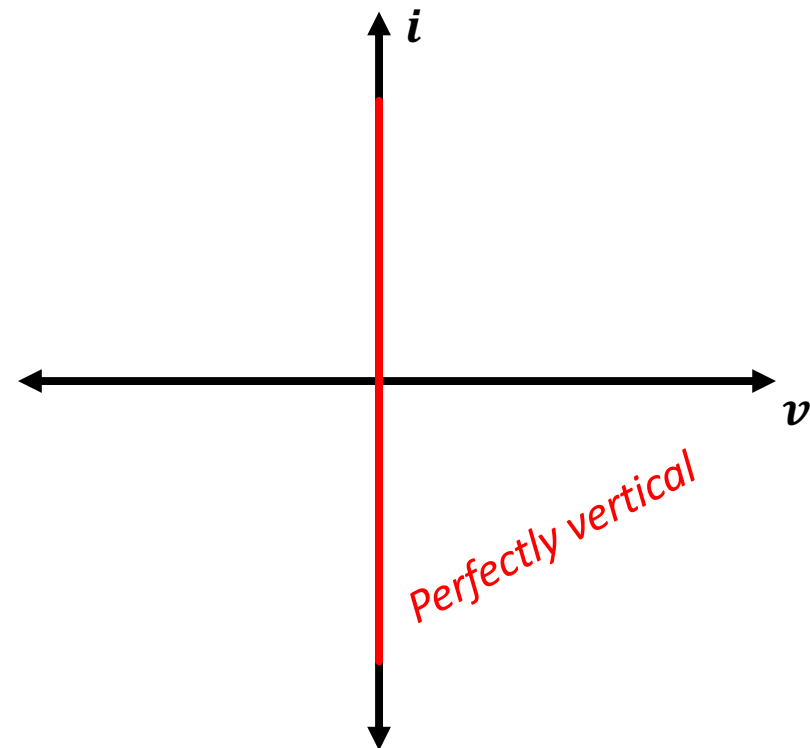
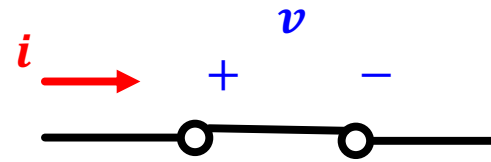


An Ideal Switch I-V Relationship

Open Switch:



Closed Switch:



Reality Though...

- Transistors Do Not Act Like that

ively (once for a given nanometer technology) [25].

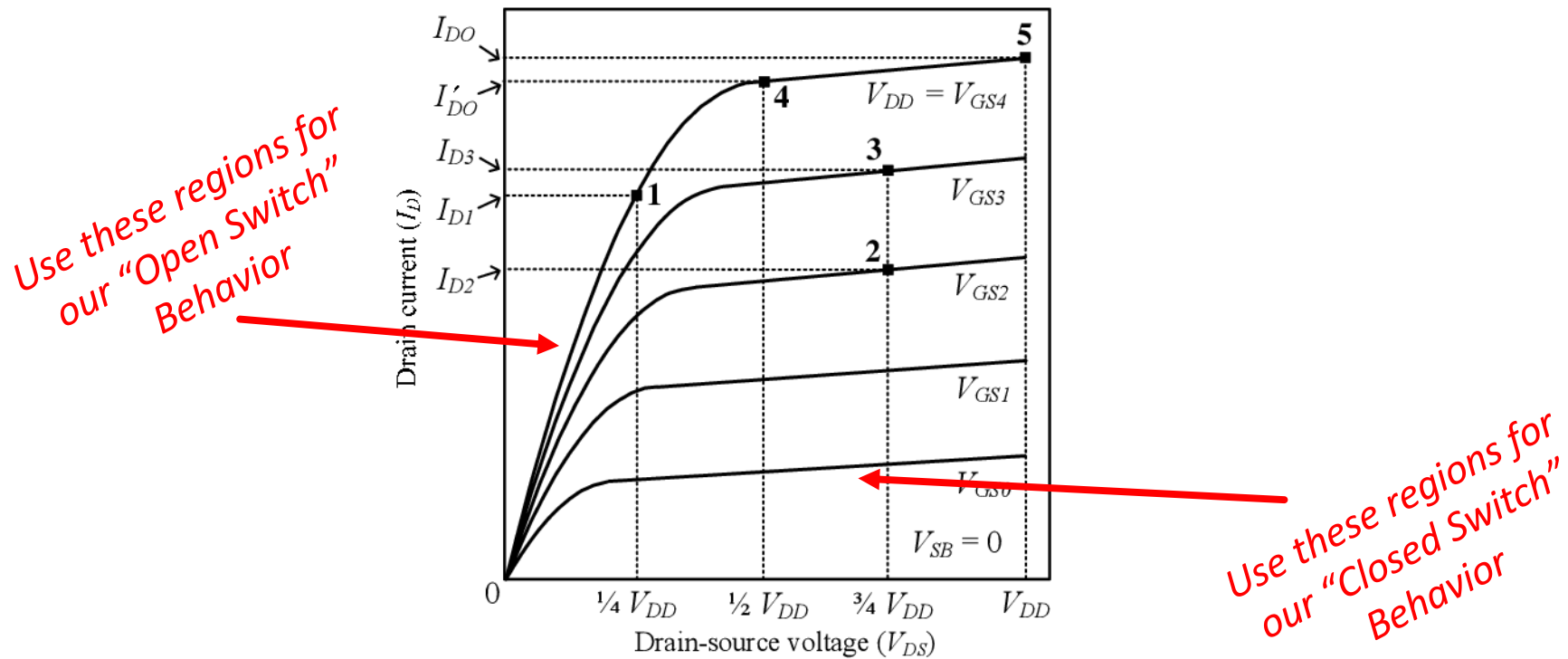
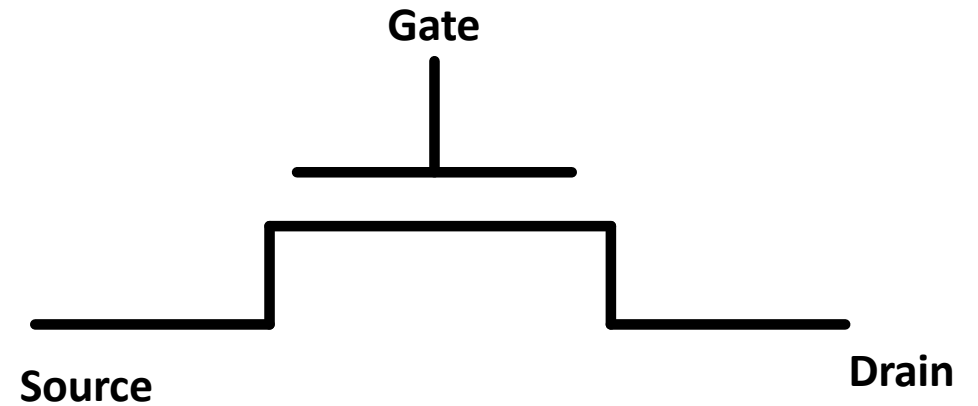


Figure 1. Selected points for NMOS parameters extraction

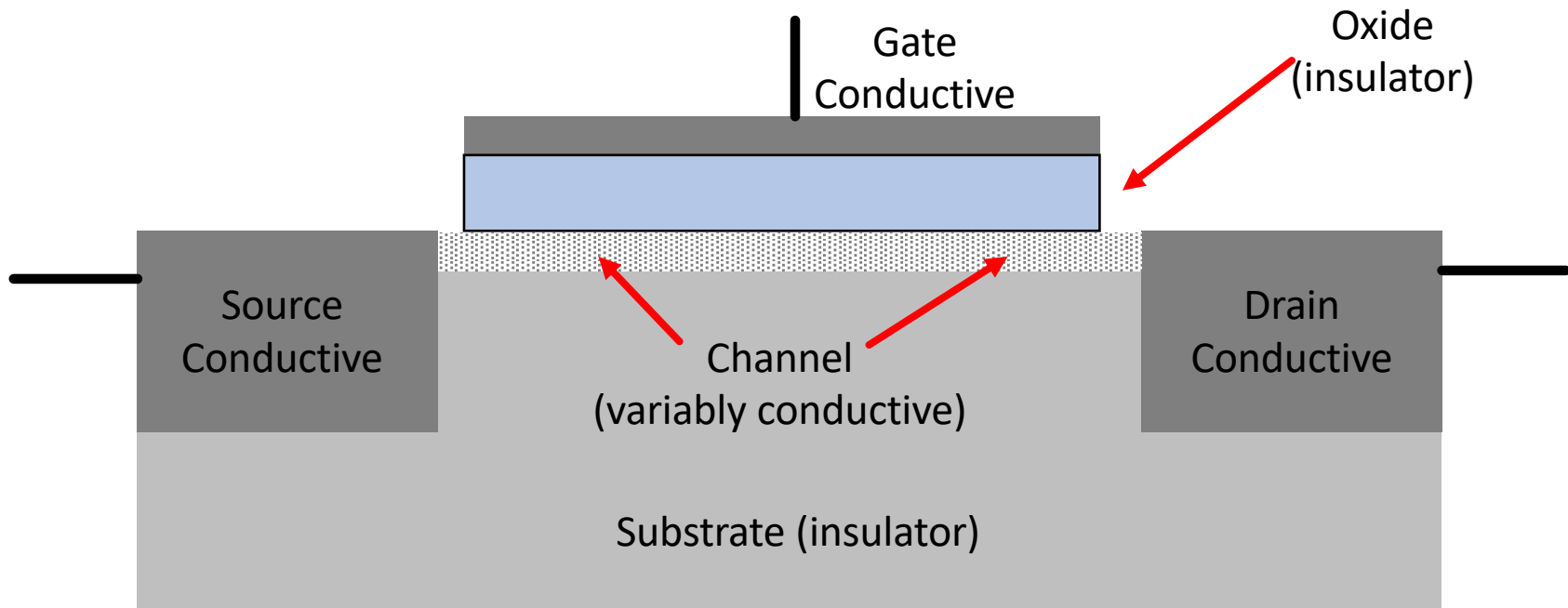
<https://www.semanticscholar.org/paper/An-Accurate-and-Compact-MOSFET-I-V-Model-for-CMOS-Bisdounis/edcd356940b102e895b83ad2e7ebcbca499710e0>

Also...A MOSFET

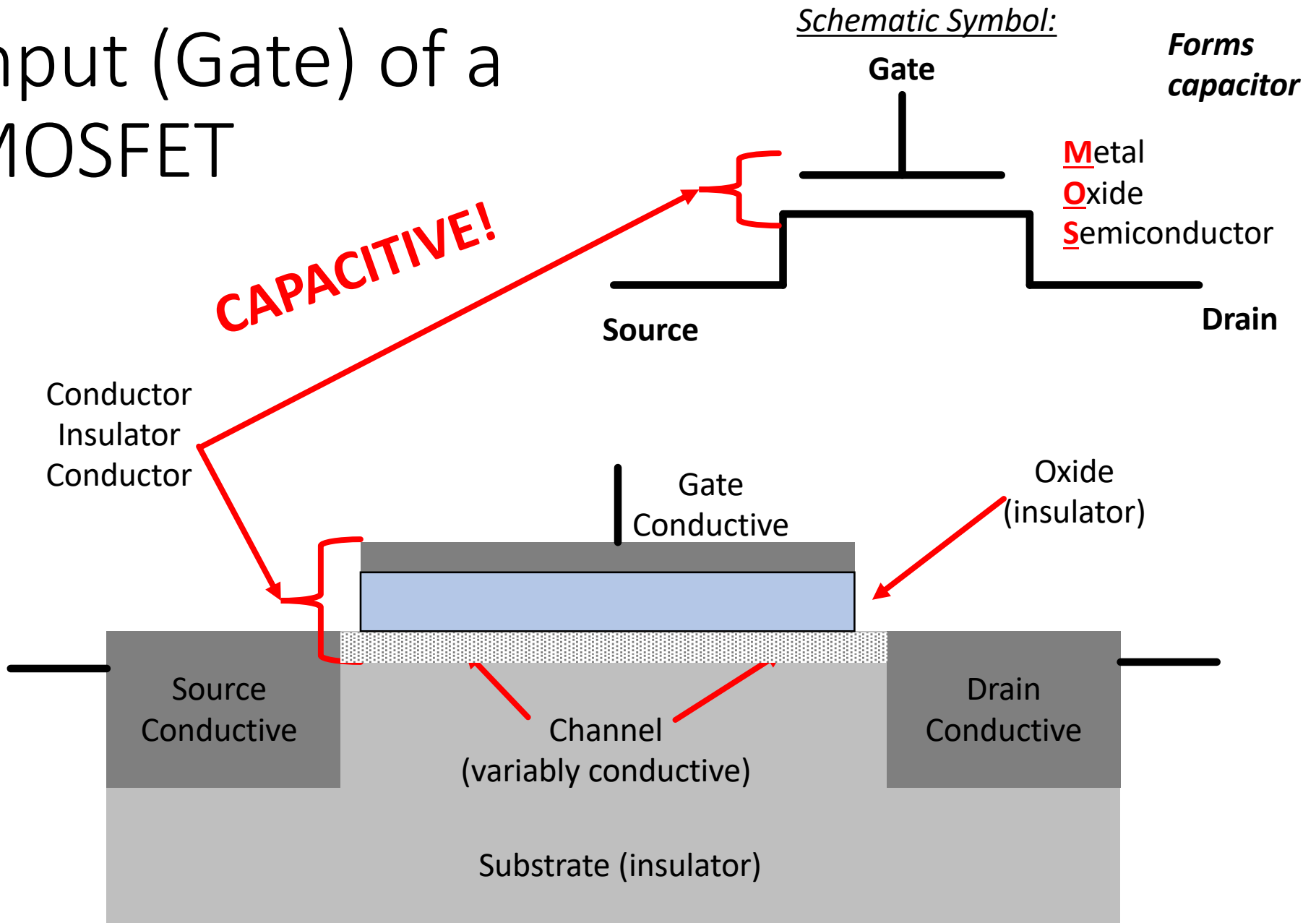
Schematic Symbol:



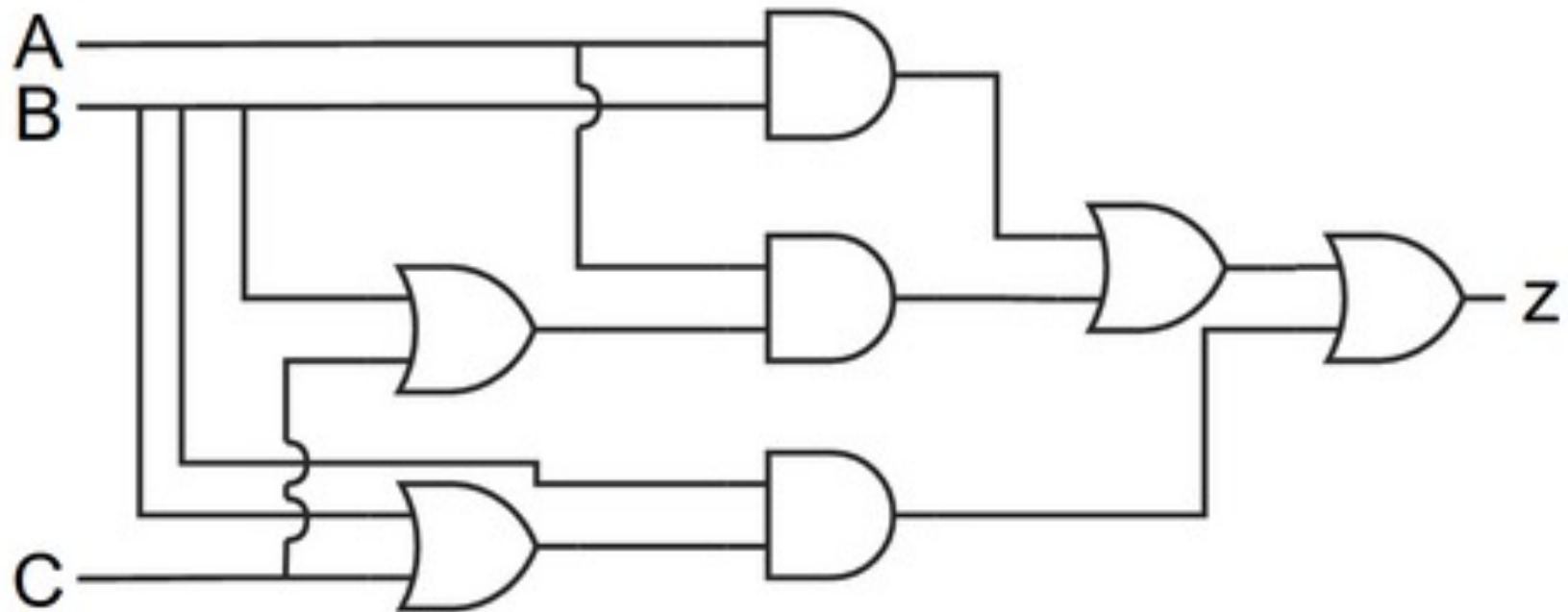
Physical Design:



Input (Gate) of a MOSFET

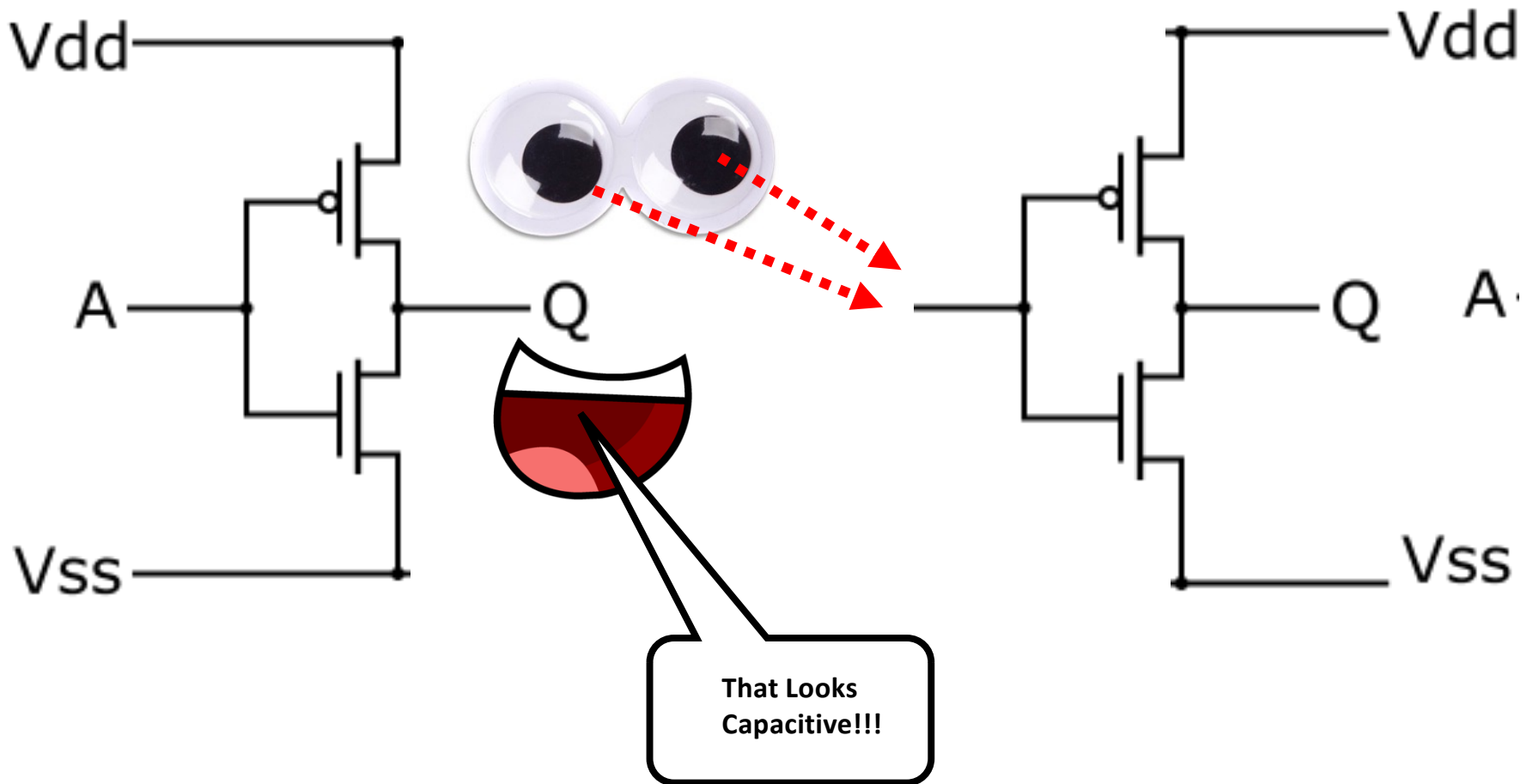


So when we start to build complicated digital circuits...



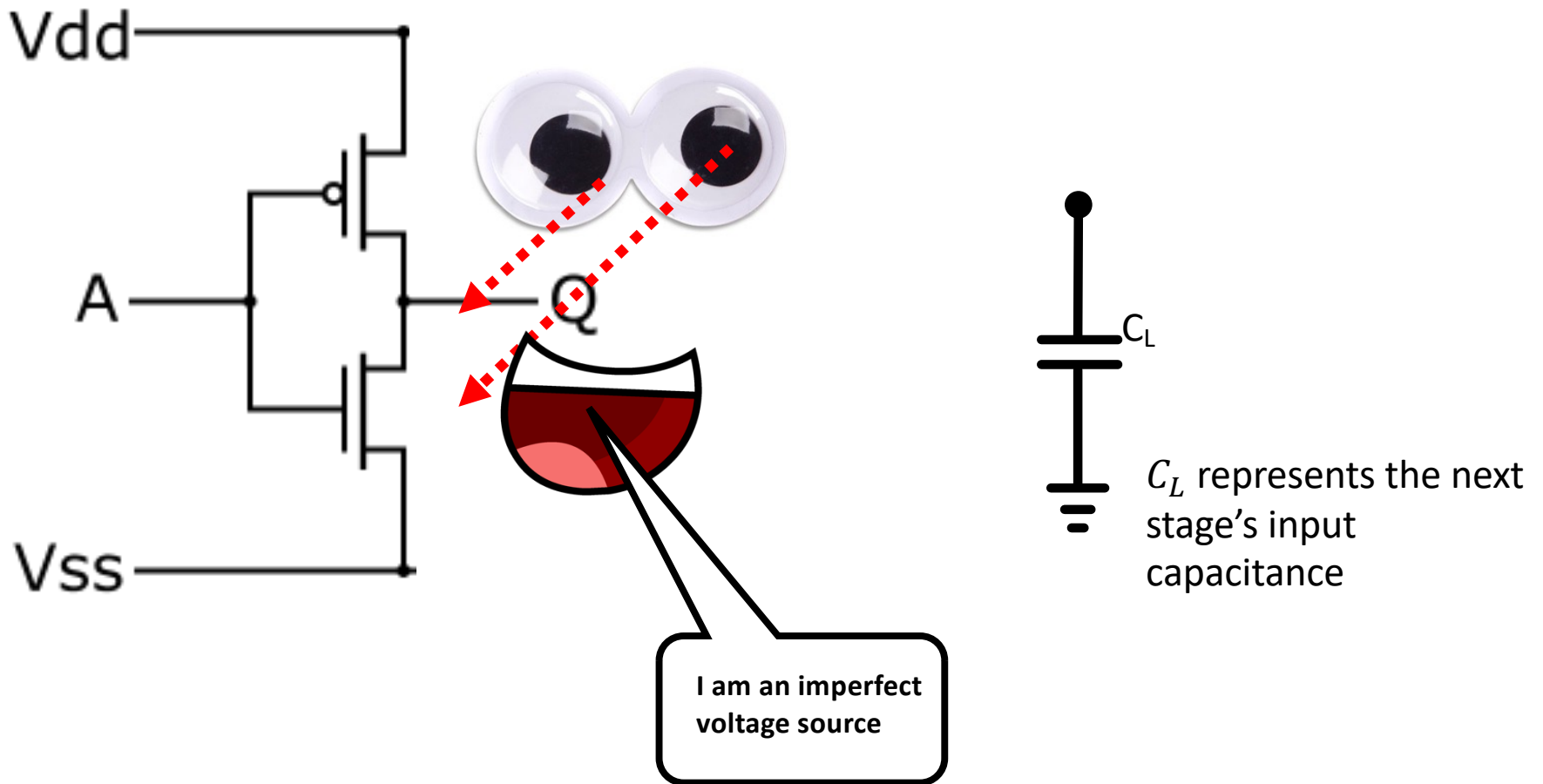
CMOS Circuit drives another CMOS Circuit...

It is CMOS all the way down



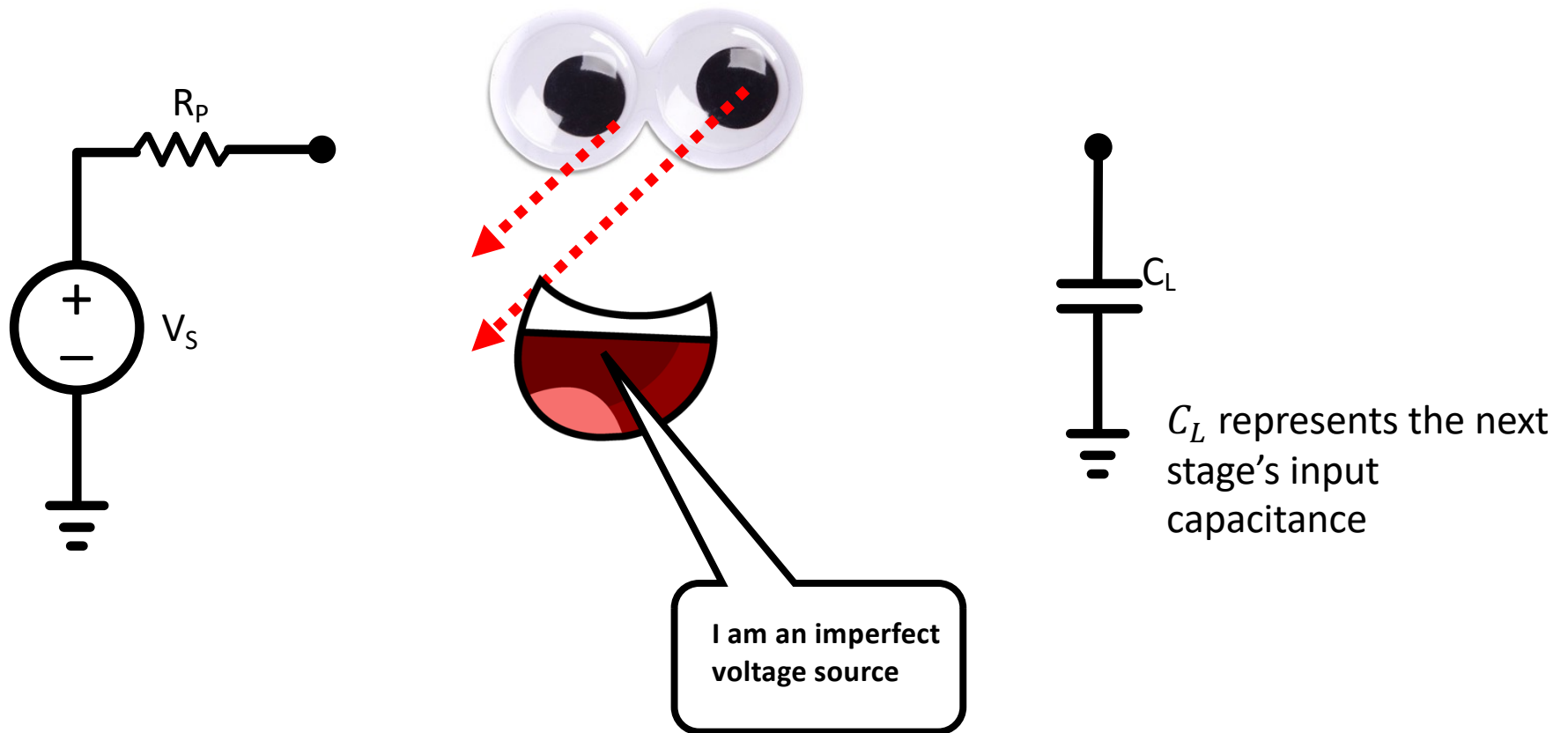
CMOS Circuit drives another CMOS Circuit...

It is CMOS all the way down



CMOS Circuit drives another CMOS Circuit...

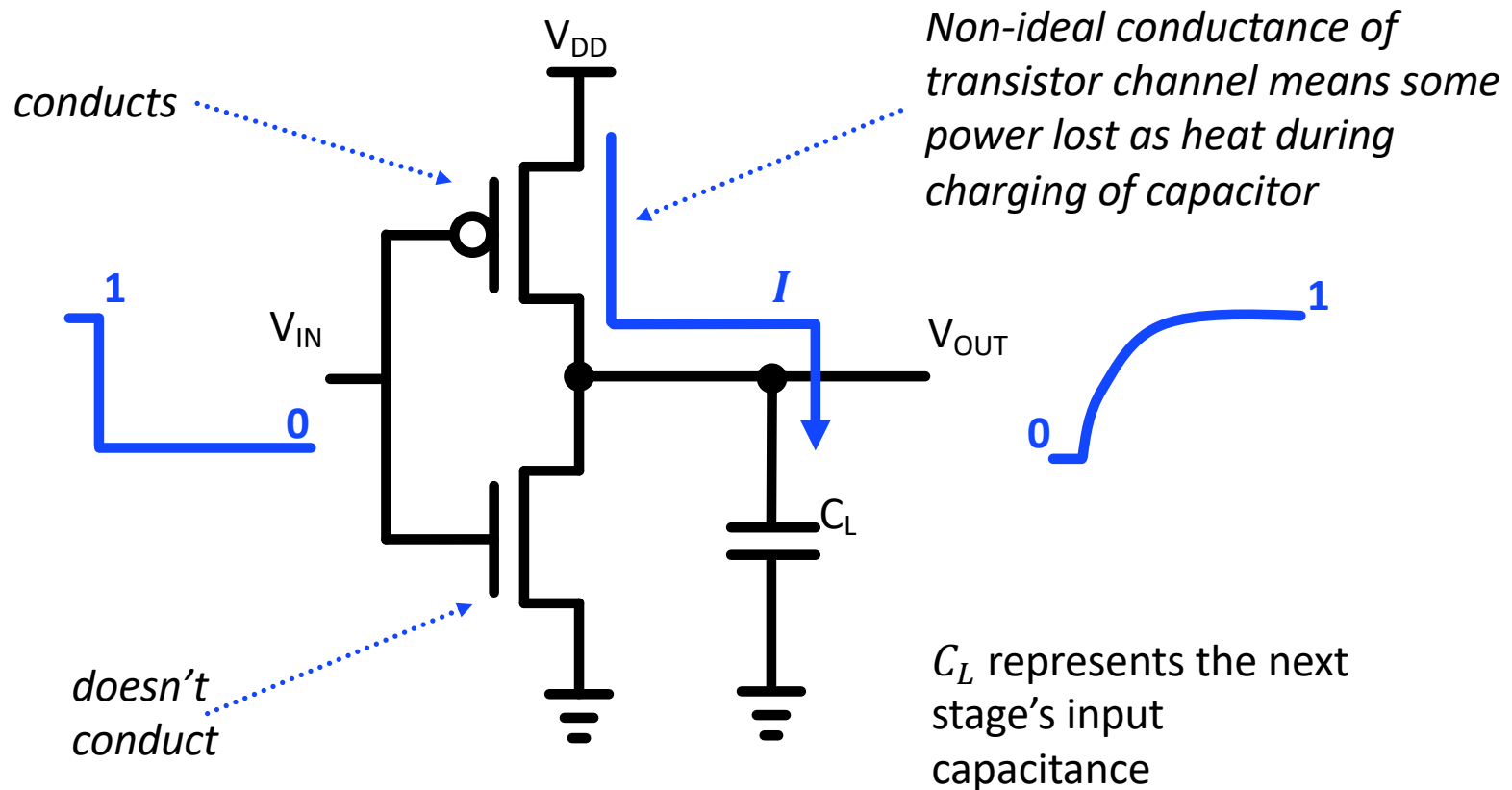
It is CMOS all the way down



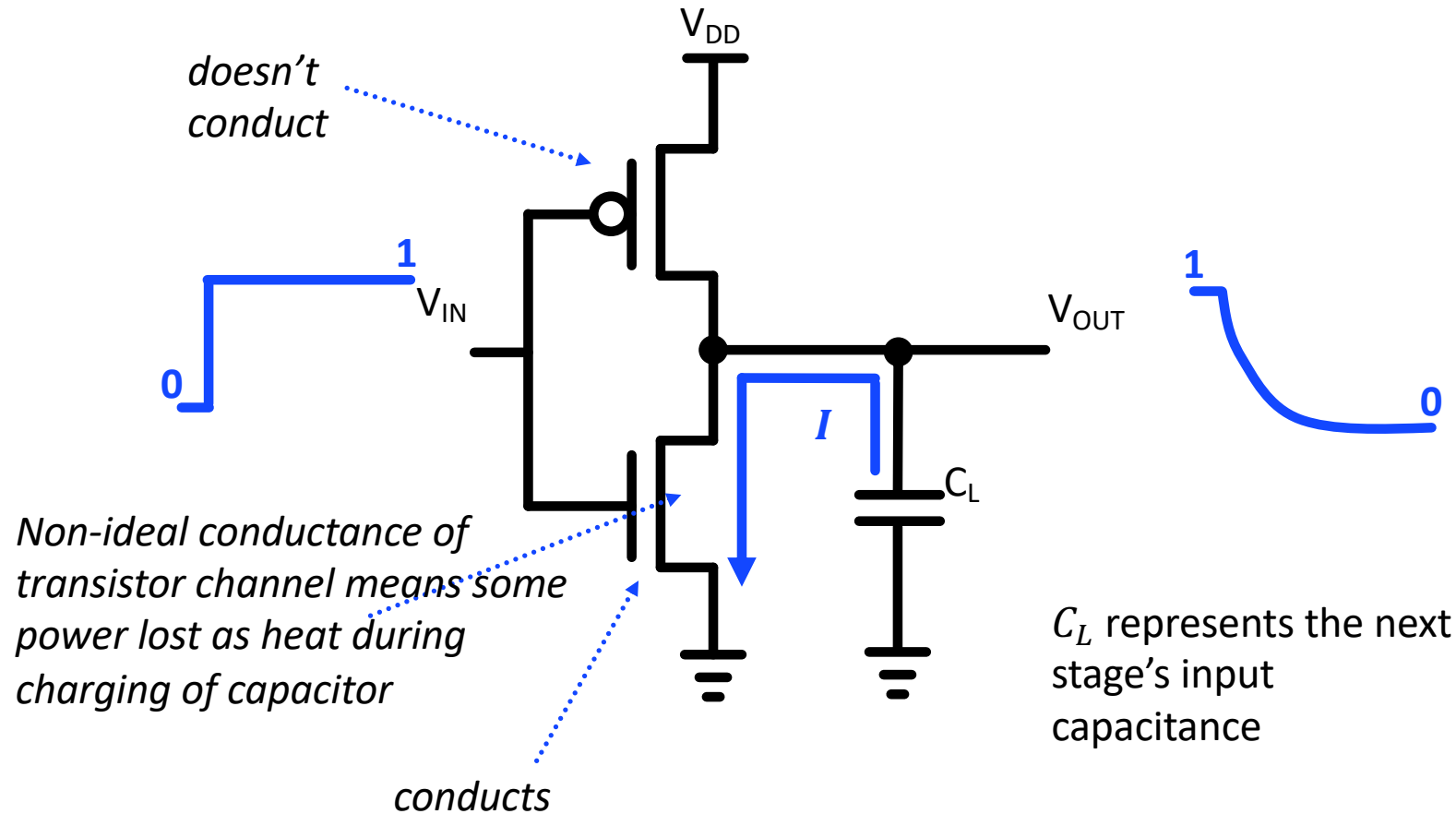
Implications

- If a circuit has to charge or discharge up a capacitive input that means...:
- There is an inherent speed limit from RC time constants that will limit the switching speed
- The fact that you have to move charge onto that capacitor means energy is needed...and that energy will never get recovered.

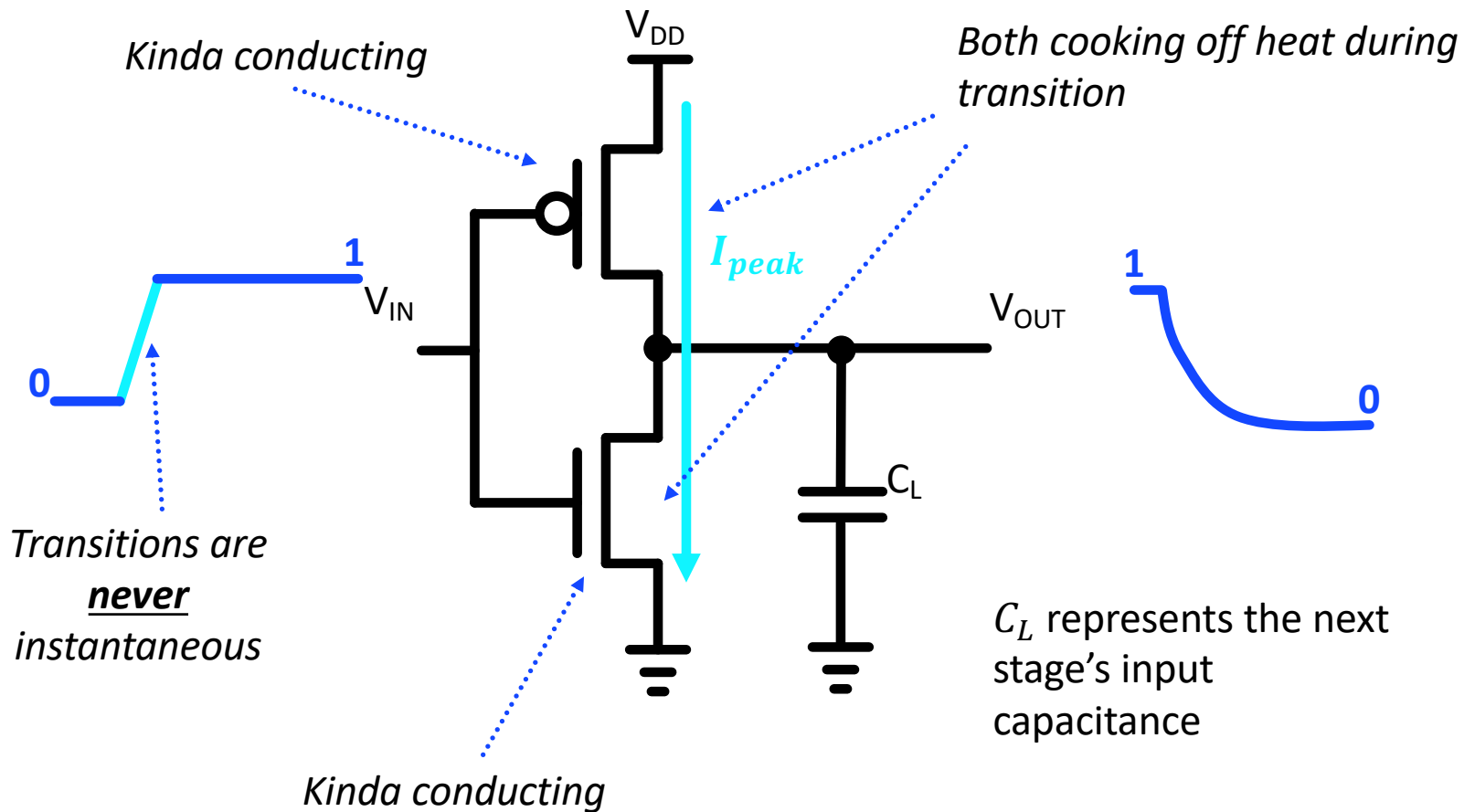
Takes energy to charge up capacitors (Dynamic Power Consumption)



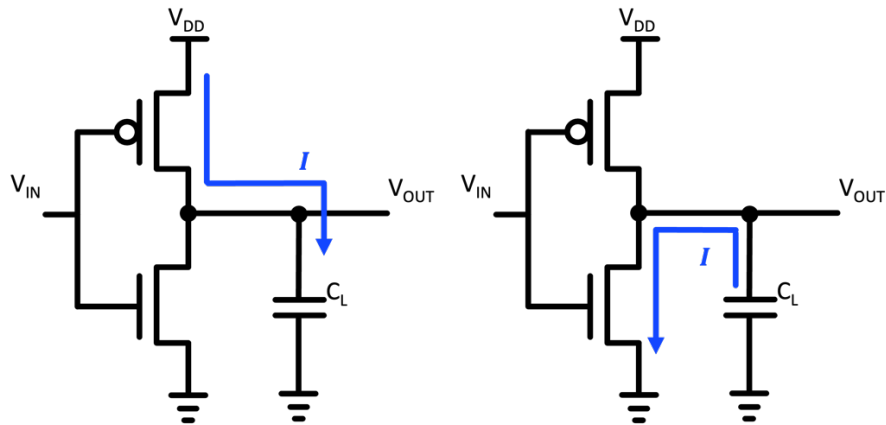
Takes energy to charge down capacitors (Dynamic Power Consumption)



During Transition, there may be overlap of conductances

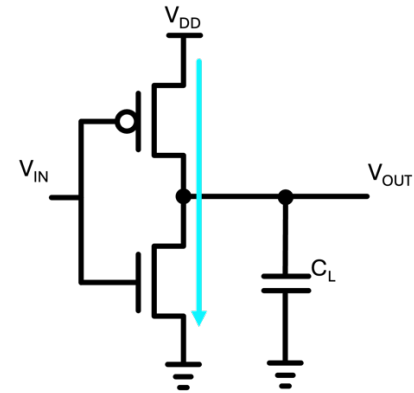


Dynamic Power Consumption



Capacitive Dis/Charging

- Caused by need to store up finite charge
- $P \propto CV^2f$
- C: capacitance of gate
- V: V_{DD} of system
- f: frequency of switching

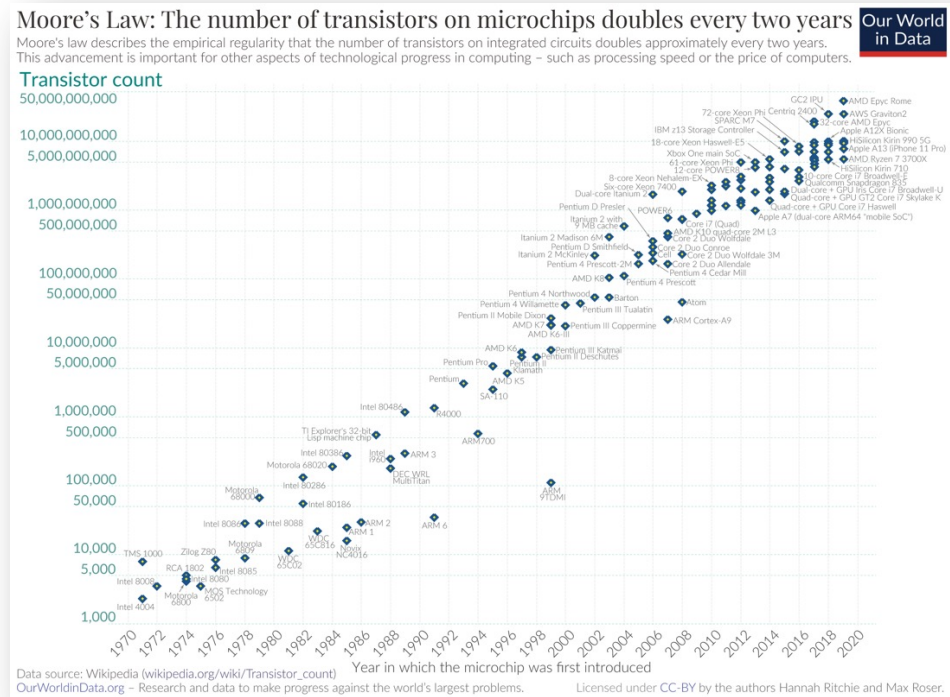


Short Circuit

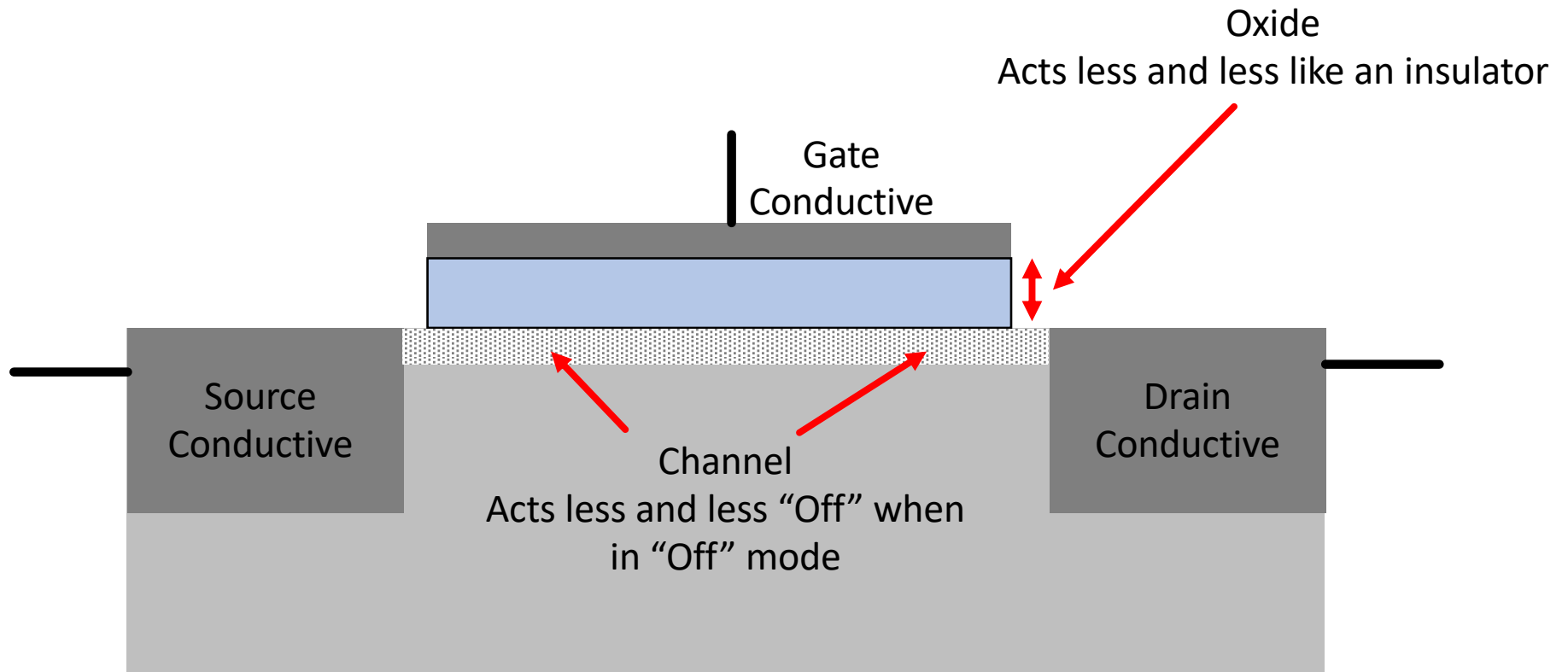
- $P \propto t_{sc}VI_{peak}$
- t_{sc} : in crossover
- V: V_{DD} of system
- I_{peak} : Max current at crossover
- Good news this is usually rather small compared to **capacitive**

The Downward Scaling of Transistors

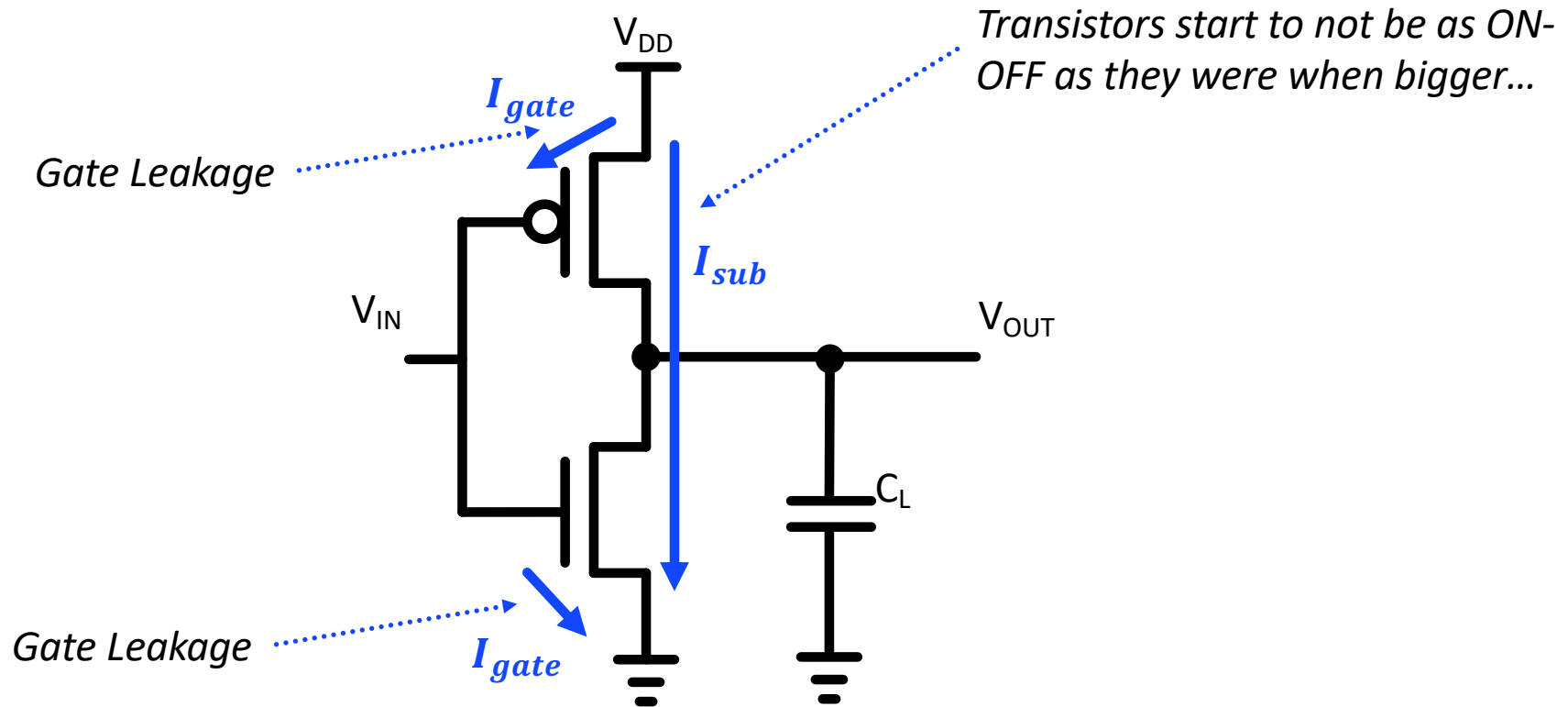
- Most gains in computation have arisen from making smaller transistors...
- Fit more on a chip.
- They use less power (less gate capacitance!!!)
- Can do more for less!



As MOSFETs get Very Small...



Static Power Consumption



Static Power Consumption

- **Gate Leakage:** The gates don't act like perfect insulators so you leak current (power) through them
- **Sub-threshold Leakage:** The transistors don't turn On and off as sharply, so there's more mushing and overlap and start to conduct significant amounts all the time (wasted leaks)

Static Power Losses

- As opposed to dynamic power losses which arise from the act of switching bits...
- Static losses take place just by being powered up and existing.
- So the total power consumed ends up being roughly explained by this equation:

$$P_{total} = P_{dynamic} + P_{static}$$

Two Major Ways Power is Consumed on a Device

- **Dynamic Power Consumption:** By flipping bits in the process of calculations, you will be burning energy. The more bits flipped, the more energy burned.
- **Static Power Consumption:** Just by “being on” you’re eating power...even if you’re not doing anything.

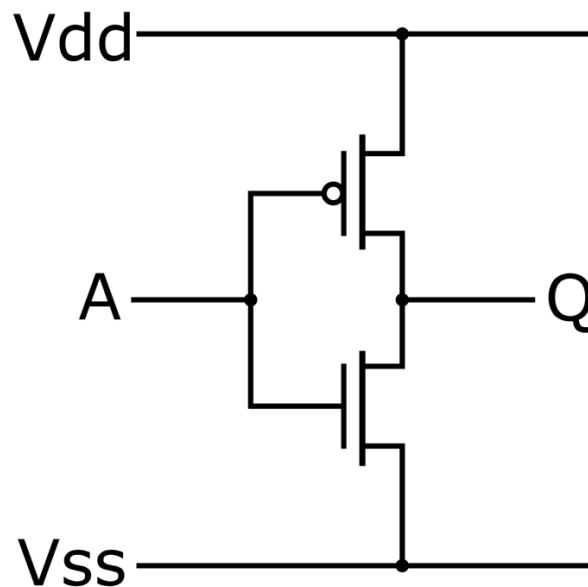
Summary: Digital Power Consumption

- P : total power consumed
- $\alpha_{0 \rightarrow 1}$: fraction of gates switching
- C : Capacitance of gates, busses, interconnects
- V : Operating voltage (V_{dd})
- f : frequency of operation
- I_{leak} : Leakage Current:
 - Sub-threshold leakage
 - Gate-Leakage

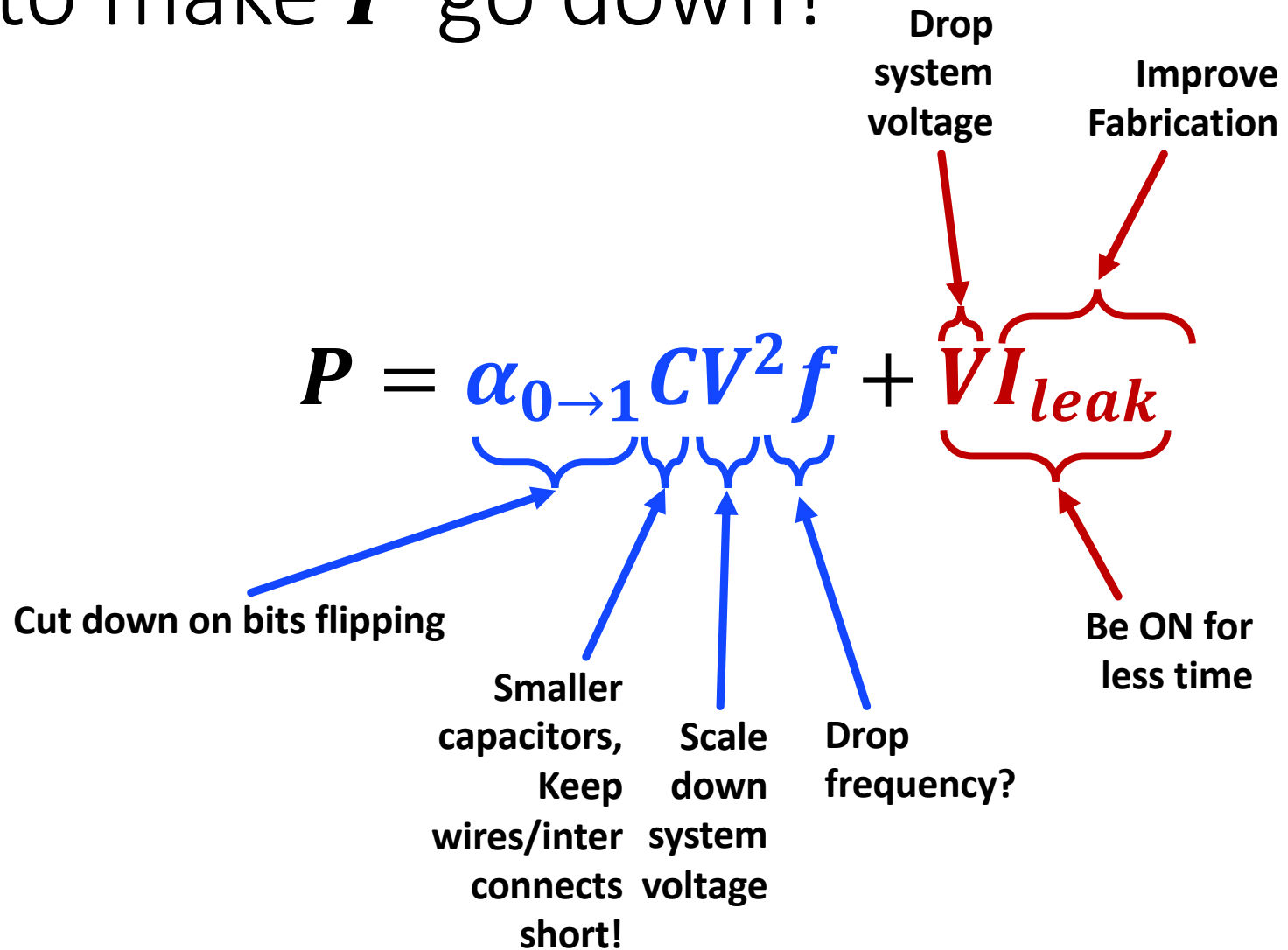
Dynamic power consumption

$$P = \alpha_{0 \rightarrow 1} CV^2 f + VI_{leak}$$

Static power consumption



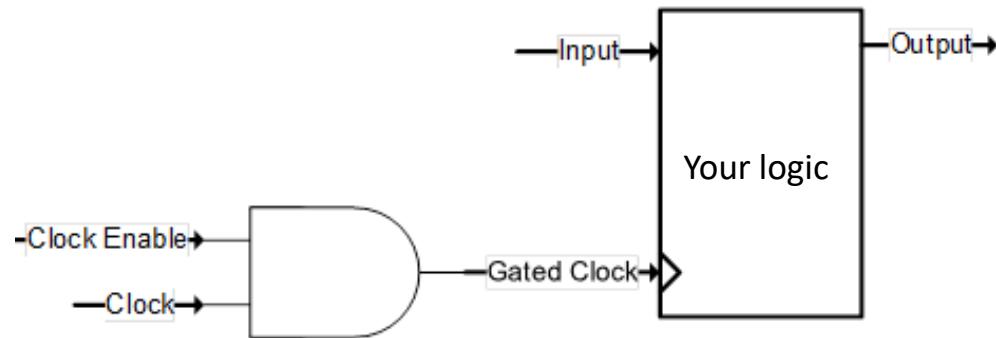
How to make P go down?



How to Manage?

- If you're designing chips from scratch you have some options:
- **Dynamic Power Consumption:**
 - **Clock Gating**: Stop the clock that is going to your logic...this will essentially “freeze” the state of your module and prevent it from running in place
- **Static Power Consumption:**
 - **Power Gating**: Actually turn off your logic...this will basically remove it from your system and not bleed power by just being on.

Clock-Gating



- Since most modern digital logic is synchronous and only updates on the edges of clocks (sequential logic), simply stopping the clock (making its period go to infinity) can “freeze” the system
- The system will not flip bits for no reason
- Relatively easy to implement into chip design

Power-Gating

- More involved than clock gating! You actually deactivate whole portions of the circuit (have large high side or low side enable transistors)
- Takes up a lot more real estate on chip
- Requires a lot more devices to make work...Have to:
 - Isolate the “dead” portion from “live portions”
 - Save and reload state of system before after turnoff
 - Worry about pipelining and propagation of removal/reappearance
 - Longer shutdown and startup times
- But does minimize power wasted!!!!

<https://anysilicon.com/power-gating/>

2/25/25

6.9000 Spring 2025

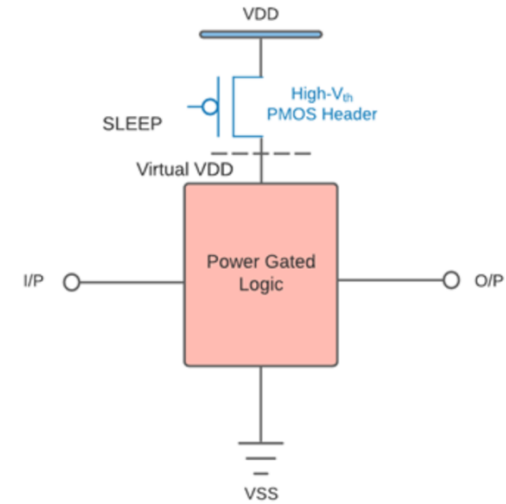


Fig. 1.3: Header Switch Cell

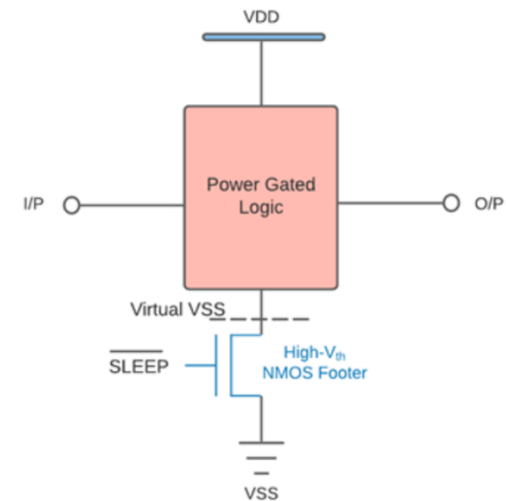


Fig. 1.4: Footer Switch cell

Both Functionalities Are Available To Us in 6.9000!

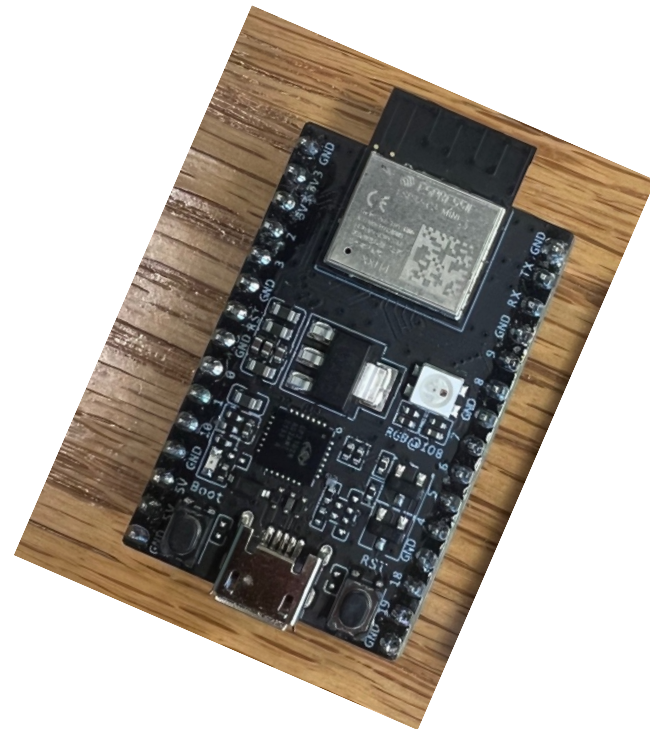
- The ESP32C3 is, if nothing else, a very power-conscious microcontroller.
- It has both clock gating and power gating of various peripherals that are at your fingertips.
- We need to take advantage of these whenever possible to save on energy

More Generally

- Try to minimize:
 - Time it takes to do something
 - Time we're "ON"
 - If we're "ON" make sure we're doing as much as possible
 - No lollygagging around.

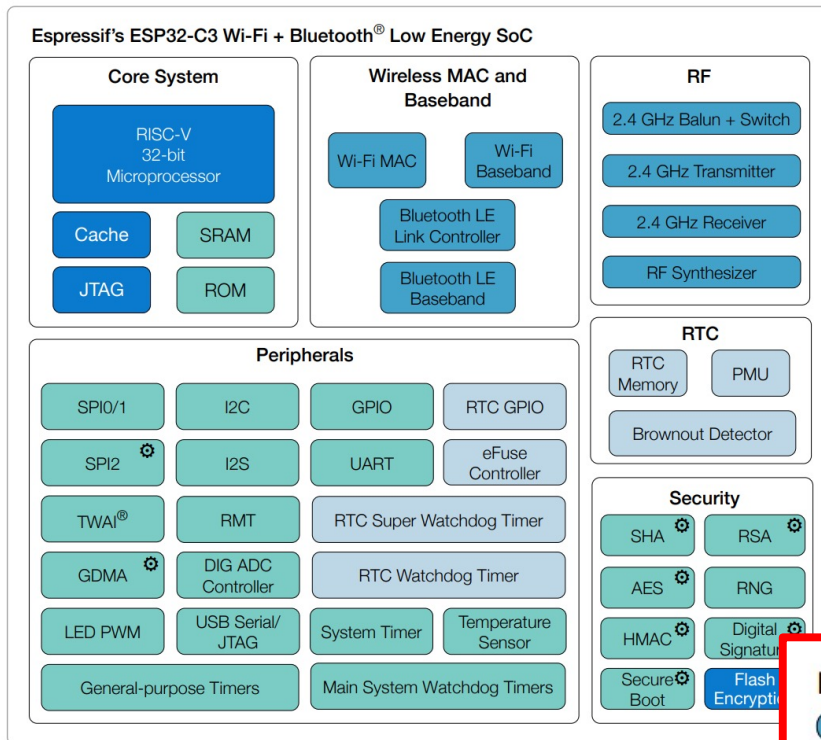
Espressif and ESP32 C3

- Tons of Documentation
- Quite a lot of functionality



ESP32 C3 Power Modes

- Various Portions of the ESP32 C3 SOC can be turned on/off as needed



Work mode	Description	Peak (mA)	
Active (RF working)	TX	802.11b, 1 Mbps, @20.5 dBm	345
		802.11g, 54 Mbps, @18 dBm	285
		802.11n, HT20, MCS7, @17.5 dBm	280
		802.11n, HT40, MCS7, @17 dBm	280
	RX	802.11b/g/n, HT20	82
		802.11n, HT40	84

Mode	CPU Frequency (MHz)	Description	Typ	
			All Peripherals Clocks Disabled (mA)	All Peripherals Clocks Enabled (mA) ¹
Modem-sleep ^{2,3}	160	CPU is idle	16	21
		CPU is running	23	28
	80	CPU is idle	13	18
		CPU is running	17	22

Mode	Description	Typ (μA)
Light-sleep	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance	130
Deep-sleep	RTC timer + RTC memory	5
Power off	CHIP_EN is set to low level, the chip is powered off	1

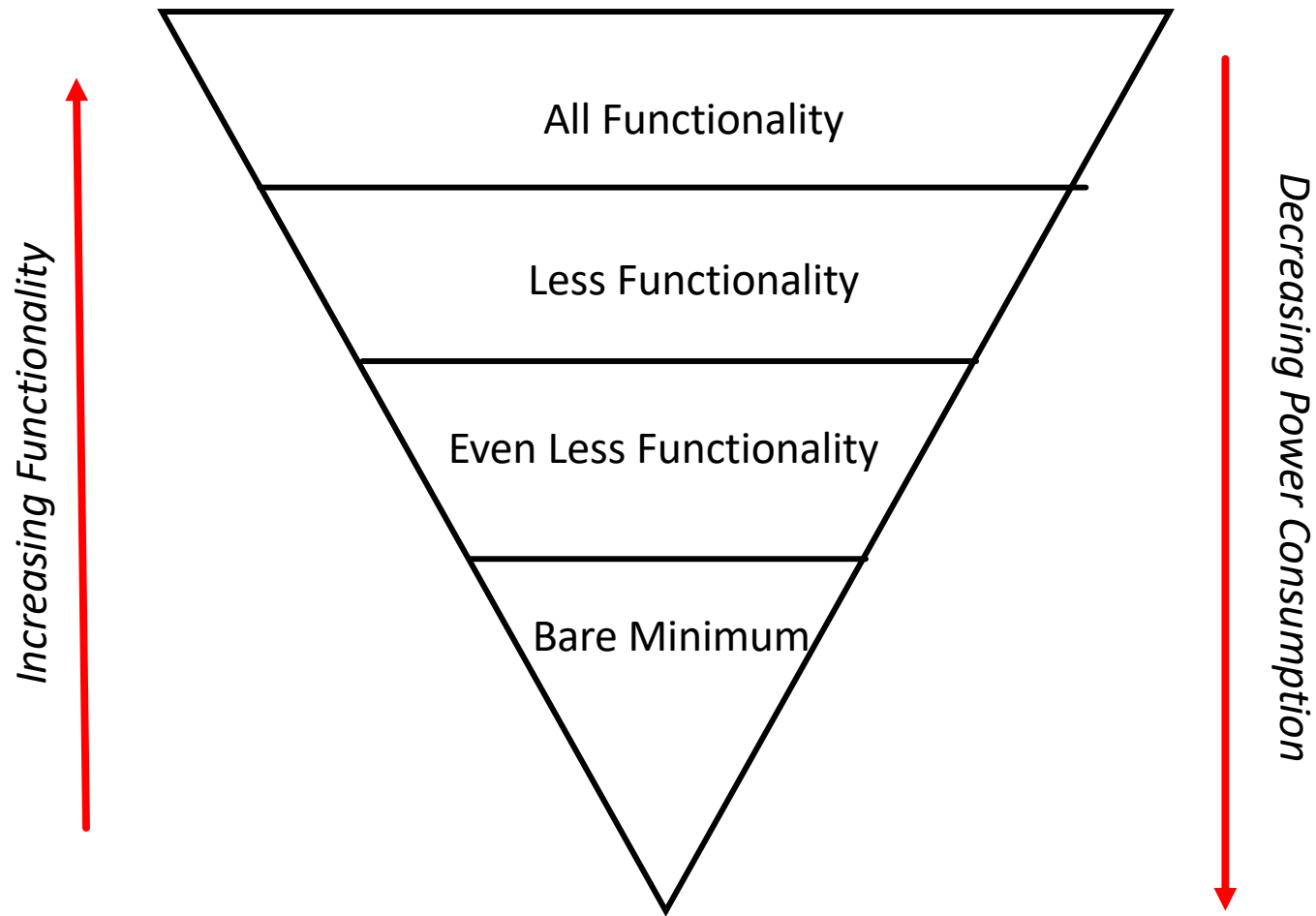
Modules having power in specific power modes:

- Active
- Active and Modem-sleep
- Active, Modem-sleep, and Light-sleep; optional in Light-sleep
- All modes

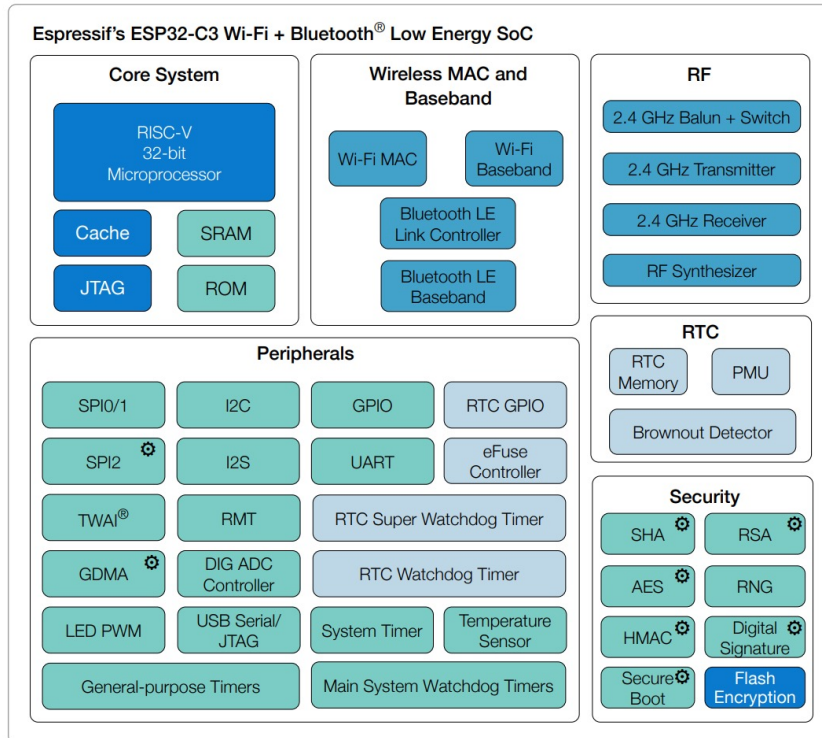
Modules having power in specific power modes:

- Active
- Active and Modem-sleep
- Active, Modem-sleep, and Light-sleep; optional in Light-sleep
- All modes

Real Hierarchy of Needs Here



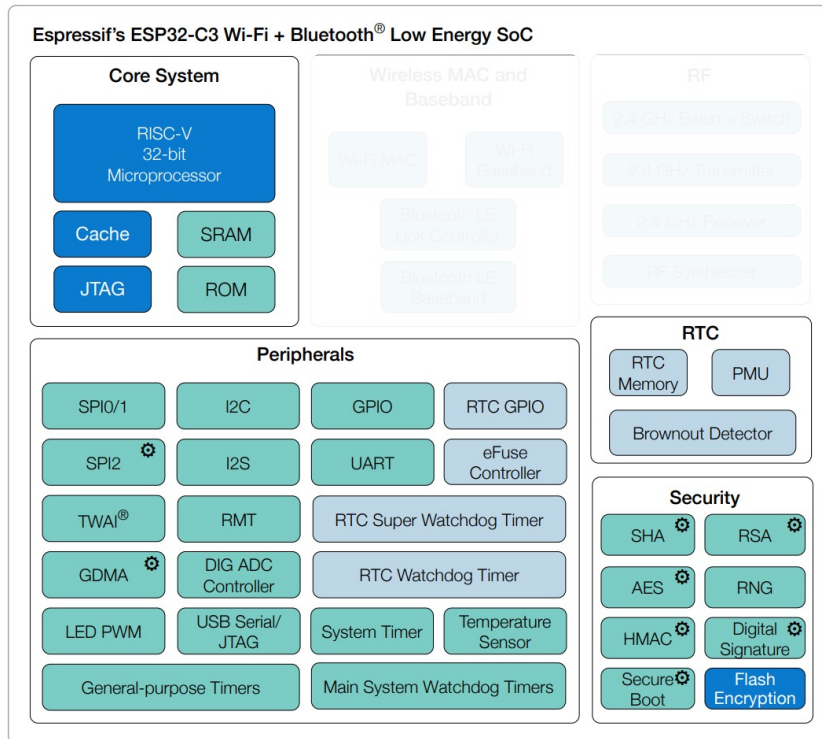
What is On/Off During “Active”



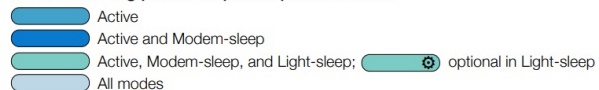
- Refers to having WiFi and/or BLE enabled
- Everything is running
- Consumes massive amount of power in various stages (TX uses more than RX...but do need to TX in order to RX)

Work mode	Description	Peak (mA)	
Active (RF working)	TX	802.11b, 1 Mbps, @20.5 dBm	345
		802.11g, 54 Mbps, @18 dBm	285
		802.11n, HT20, MCS7, @17.5 dBm	280
		802.11n, HT40, MCS7, @17 dBm	280
	RX	802.11b/g/n, HT20	82
		802.11n, HT40	84

What is On/Off During Modem-Sleep

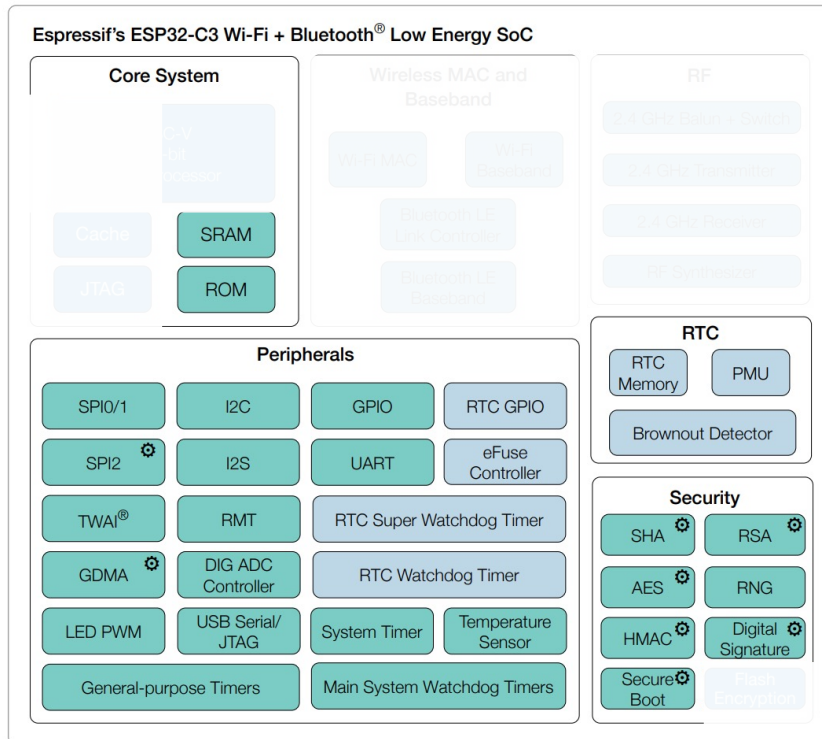


Modules having power in specific power modes:



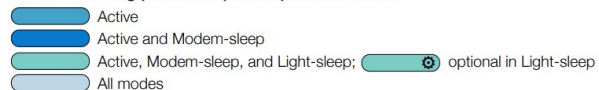
- All RF portions are turned off
- Device is basically just a microcontroller at that point

What is On/Off During Light Sleep



- Everything listed so far
- The processor is off
- Cache is off
- JTAG is off

Modules having power in specific power modes:



Light Sleep Really messes with USB

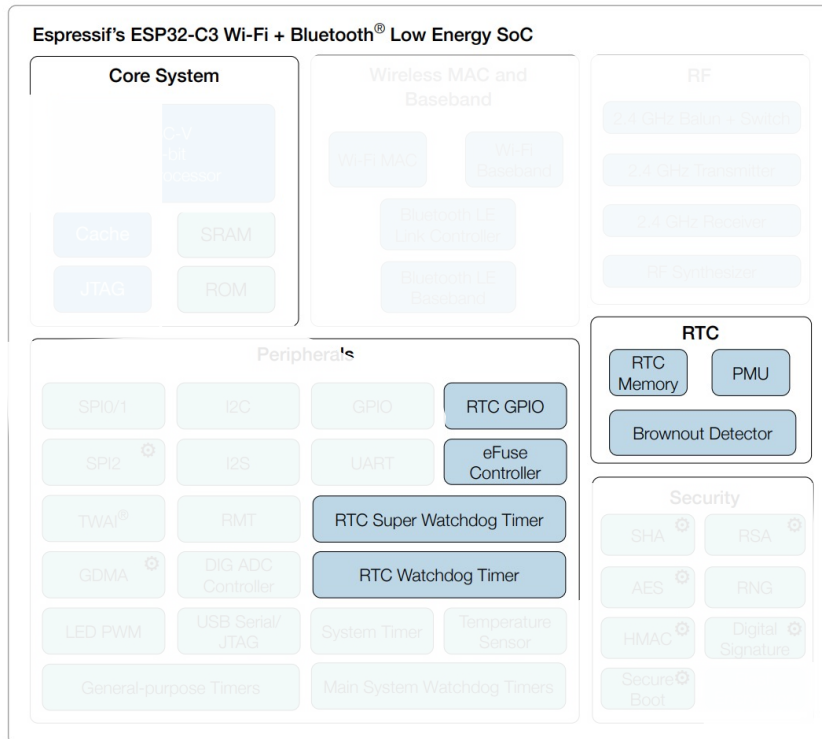
Light Sleep

When entering light sleep, the APB and USB PHY **clock are gated**. Thus, the USB Serial/JTAG controller is no longer able to receive or respond to any USB transactions from the connected host (including periodic CDC Data IN transactions). As a result:

- when entering light sleep, the USB Serial/JTAG device is unresponsive to the host/PC's USB CDC driver. The host/PC may then report the USB Serial/JTAG device as disconnected or erroneous (even if the USB cable is still physically connected).
- when exiting light sleep, it is possible that the host/PC does not re-enumerate (i.e., reconnect) the USB Serial/JTAG device given that the USB PHY's D+ line remains pulled up state during light sleep. Users may need to physically disconnect and then reconnect the USB cable.

- In my experience, with light sleep the USB port does not work after waking again...your computer gets confused and you need to reset ESP32/restart it.
- Clock gating makes USB look “frozen” to host/computer

What is On/Off During Deep Sleep



Modules having power in specific power modes:

- Active
- Active and Modem-sleep
- Active, Modem-sleep, and Light-sleep; optional in Light-sleep
- All modes

- Everything listed so far
- The processor is off
- Cache is off
- JTAG is off
- Almost All Peripherals
- Just a minimum set of functionalities that can carry you forward

The Mammalian Dive Reflex

- When any mammal holds its breath and has its sinuses exposed to low temperatures...
- Heart rate drops
- Vasculature in extremities contracts
- Blood flow prioritizes vital organs

- Just like an ESP32 C3 with power!

Deep Sleep Messes with USB Less

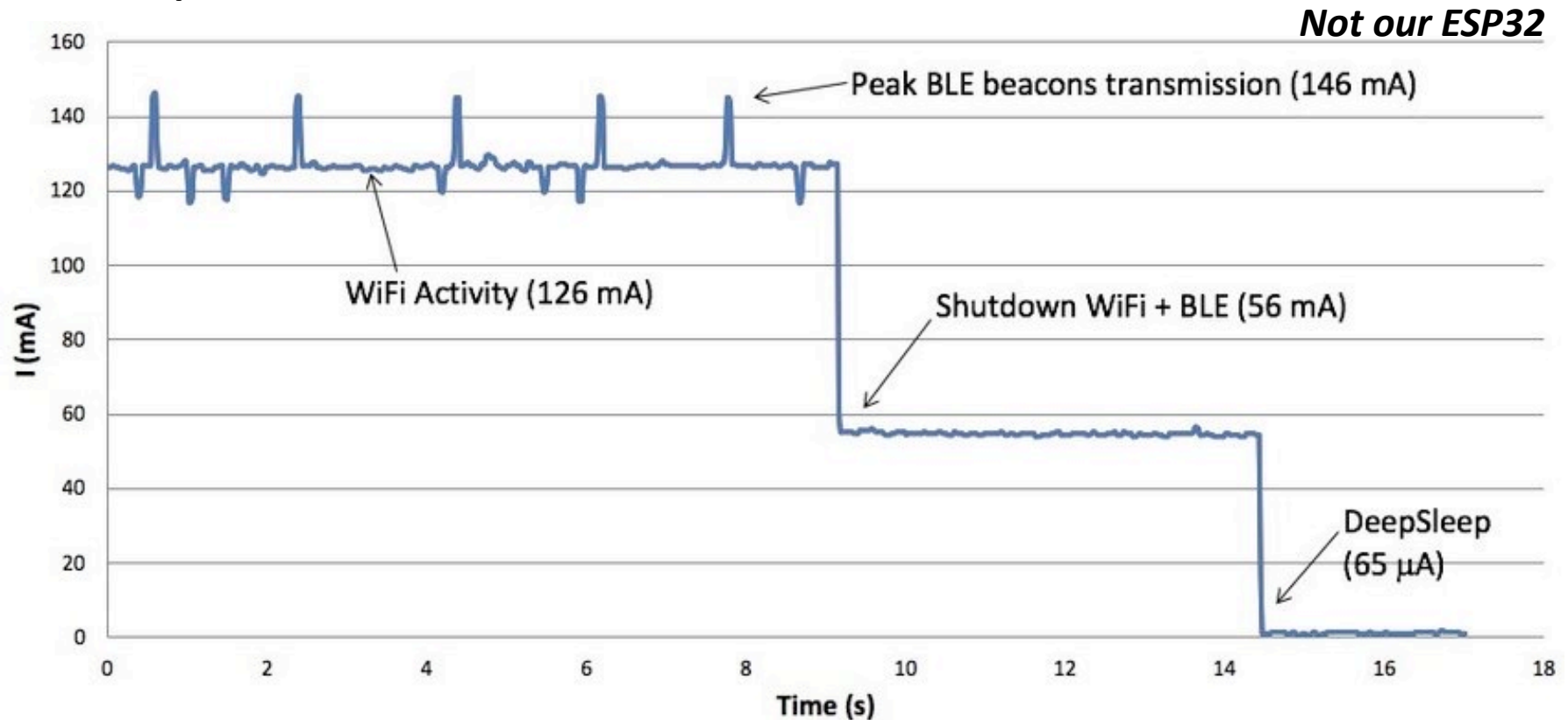
Deep Sleep

When entering deep sleep, both the USB Serial/JTAG controller and the USB PHY are powered off, leading to the USB PHY's D+ line no longer being pulled up. As a result:

- When entering deep sleep, the USB Serial/JTAG device appears disconnected from the host/PC (even if the USB cable is still physically connected).
- When exiting deep sleep, the USB Serial/JTAG device reconnects to the host/PC.

- USB doesn't work in deep sleep, but since it is powered off the computer doesn't freak out about it. Upon coming back from Deep sleep, USB controller on ESP32 C3 reboots so actually will come back up fine

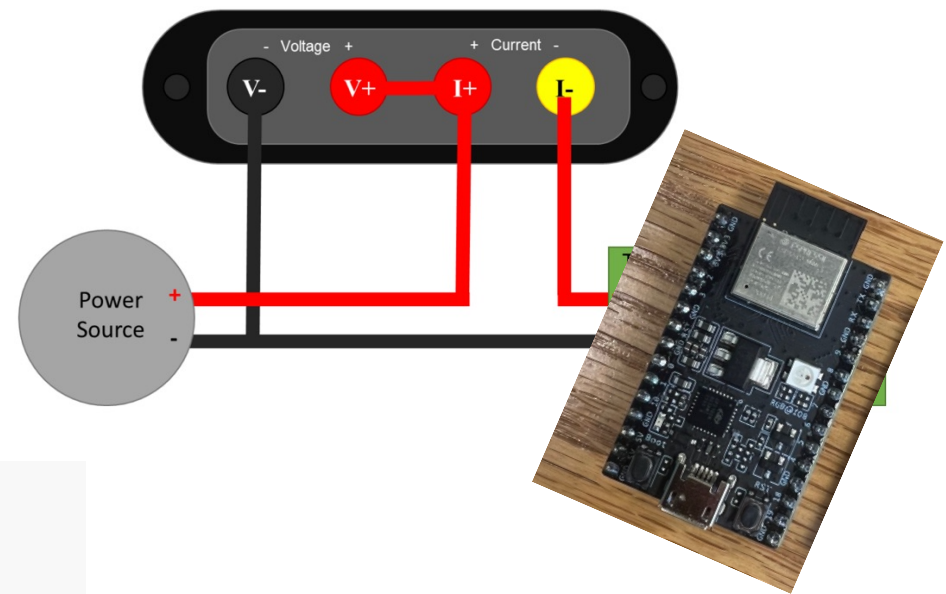
ESP32 Power Consumption is Very Complicated



<https://www.researchgate.net/figure/Energy-consumption-of-Sparkfun-ESP32-Things-fig4-332407808>

Take the JouleScope and measure power consumption in different modes

- 6.9000 Lesson: Never Trust Anybody



*Espressif is manufacturer of ESP32

ESP32 C3 Measurements (Code from Lab 01)

During regular running no wifi:



Mode	CPU Frequency (MHz)	Description	Typ	
			All Peripherals Clocks Disabled (mA)	All Peripherals Clocks Enabled (mA) ¹
	160	CPU is idle	16	21
		CPU is running	23	28
Modem-sleep ^{2,3}	80	CPU is idle	13	18
		CPU is running	17	22

Probably right???

During running with Wifi at some point during the request/contact cycle



Work mode	Description	Peak (mA)
Active (RF working)	802.11b, 1 Mbps, @20.5 dBm	345
	802.11g, 54 Mbps, @18 dBm	285
	802.11n, HT20, MCS7, @17.5 dBm	280
	802.11n, HT40, MCS7, @17 dBm	280
	802.11b/g/n, HT20	82
	802.11n, HT40	84

Probably right???

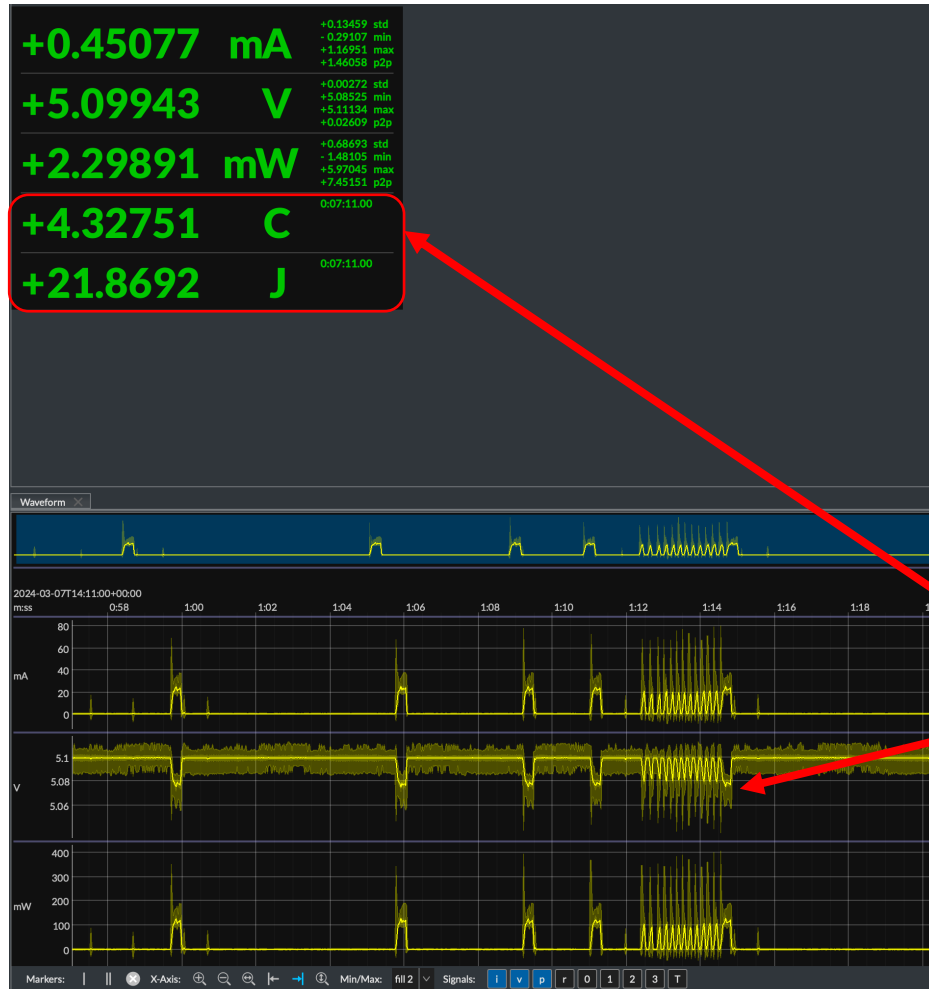
The power consumption is complicated

Connecting to WiFi

Requests to Numbers API



Worry about Power but also Energy



- The power pattern that appears during complicated tasks such as a WiFi request may not be best encompassed by an instantaneous power
- May need to worry more about energy per message
- Or something else related to integrating power over time!!!!

Must dig through docs and app notes!

Power Management

[中文]

Overview

Power management algorithm included in ESP-IDF can adjust the advanced peripheral bus (APB) frequency, CPU frequency, and put the chip into Light-sleep mode to run an application at smallest possible power consumption, given the requirements of application components.

Application components can express their requirements by creating and acquiring power management locks.

For example:

- Driver for a peripheral clocked from APB can request the APB frequency to be set to 80 MHz while the peripheral is used.
- RTOS can request the CPU to run at the highest configured frequency while there are tasks ready to run.
- A peripheral driver may need interrupts to be enabled, which means it has to request disabling Light-sleep.

Since requesting higher APB or CPU frequencies or disabling Light-sleep causes higher current consumption, please keep the usage of power management locks by components to a minimum.

Configuration

Power management can be enabled at compile time, using the option `CONFIG_PM_ENABLE`.

Enabling power management features comes at the cost of increased interrupt latency. Extra latency depends on a number of factors, such as the CPU frequency, single/dual core mode, whether or not frequency switch needs to be done. Minimum extra latency is 0.2 us (when the CPU frequency is 240 MHz and frequency scaling is not enabled). Maximum extra latency is 40 us (when frequency scaling is enabled, and a switch from 40 MHz to 80 MHz is performed on interrupt entry).

Dynamic frequency scaling (DFS) and automatic Light-sleep can be enabled in an application by calling the function `esp_pm_configure()`. Its argument is a structure defining the frequency scaling

ESP32-C3

Wireless Adventure: A Comprehensive Guide to IoT

RISC-V Wi-Fi Bluetooth ESP-IDF ESP RainMaker

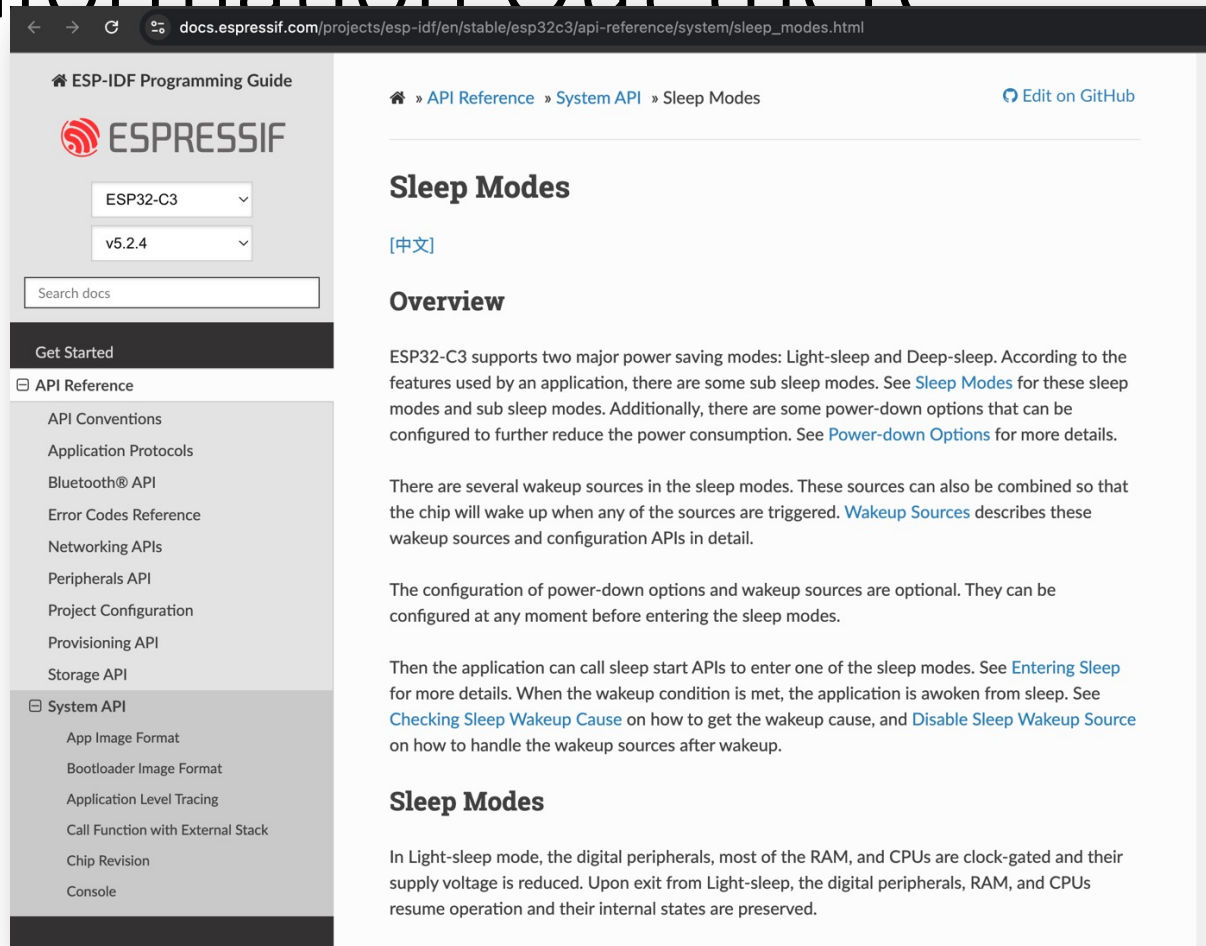
ESPRESSIF

<https://www.espressif.com/sites/default/files/documentation/ESP32-C3%20Wireless%20Adventure.pdf>

CHAPTER 12

https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/api-reference/system/power_management.html

Lots of Information Out there



The screenshot shows a web browser displaying the ESP-IDF Programming Guide. The URL is `docs.espressif.com/projects/esp-idf/en/stable/esp32c3/api-reference/system/sleep_modes.html`. The page title is "ESP-IDF Programming Guide" and the breadcrumb is "API Reference » System API » Sleep Modes". The main content area is titled "Sleep Modes" and includes a link to "[中文]". The "Overview" section states: "ESP32-C3 supports two major power saving modes: Light-sleep and Deep-sleep. According to the features used by an application, there are some sub sleep modes. See [Sleep Modes](#) for these sleep modes and sub sleep modes. Additionally, there are some power-down options that can be configured to further reduce the power consumption. See [Power-down Options](#) for more details." It also mentions: "There are several wakeup sources in the sleep modes. These sources can also be combined so that the chip will wake up when any of the sources are triggered. [Wakeup Sources](#) describes these wakeup sources and configuration APIs in detail." and "The configuration of power-down options and wakeup sources are optional. They can be configured at any moment before entering the sleep modes." The "Sleep Modes" section begins with: "In Light-sleep mode, the digital peripherals, most of the RAM, and CPUs are clock-gated and their supply voltage is reduced. Upon exit from Light-sleep, the digital peripherals, RAM, and CPUs resume operation and their internal states are preserved." A sidebar on the left contains a navigation menu with categories like "Get Started", "API Reference", and "System API".

<https://blog.voltaicsystems.com/how-to-put-an-esp32-into-deep-sleep/>

<https://randomnerdtutorials.com/esp32-timer-wake-up-deep-sleep/>

Experiments

- Wrote Some Test Files That Put the microcontroller into different modes
- This line lets the ESP wake itself from various sleep modes after a certain amount of time:

```
esp_sleep_enable_timer_wakeup(5000000); // how long to sleep in us.
```

- Then you can either go into light sleep:

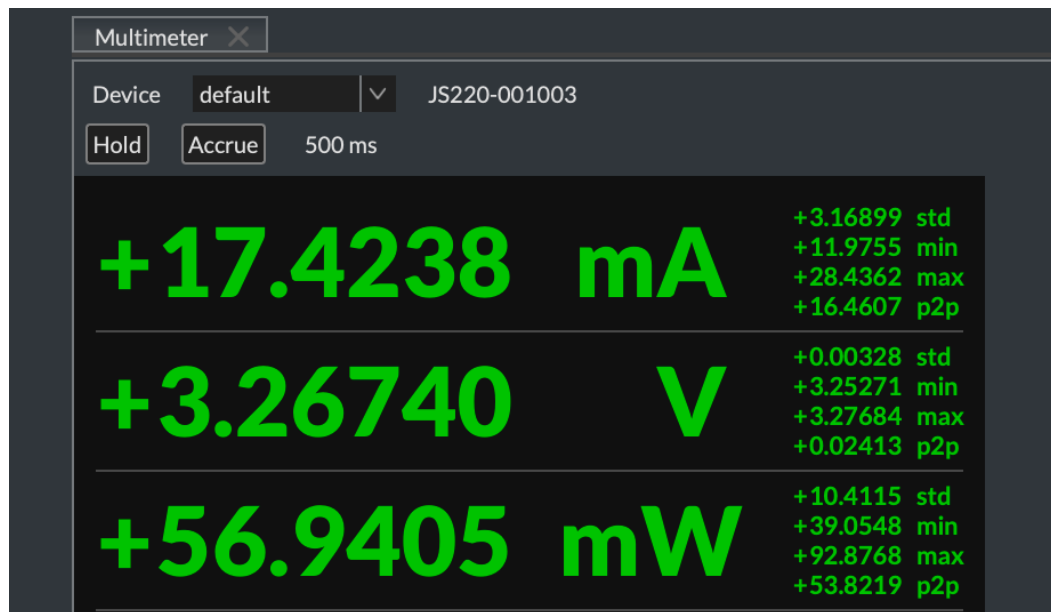
```
esp_light_sleep_start();
```

- Or Deep Sleep:

```
esp_deep_sleep_start();
```

So for reference regular running with no wifi/BLE was like...

Mode	CPU Frequency (MHz)	Description	Typ	
			All Peripherals Clocks Disabled (mA)	All Peripherals Clocks Enabled (mA) ¹
Modem-sleep ^{2,3}	160	CPU is idle	16	21
		CPU is running	23	28
	80	CPU is idle	13	18
		CPU is running	17	22



Light Sleep:

```
esp_light_sleep_start();
```

Seems to go to sleep on a timer...

Mode	Description	Typ (μ A)
Light-sleep	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance	130
Deep-sleep	RTC timer + RTC memory	5
Power off	CHIP_EN is set to low level, the chip is powered off	1



Definitely Lower Power but is it good enough?

Why is it 6 times as large?

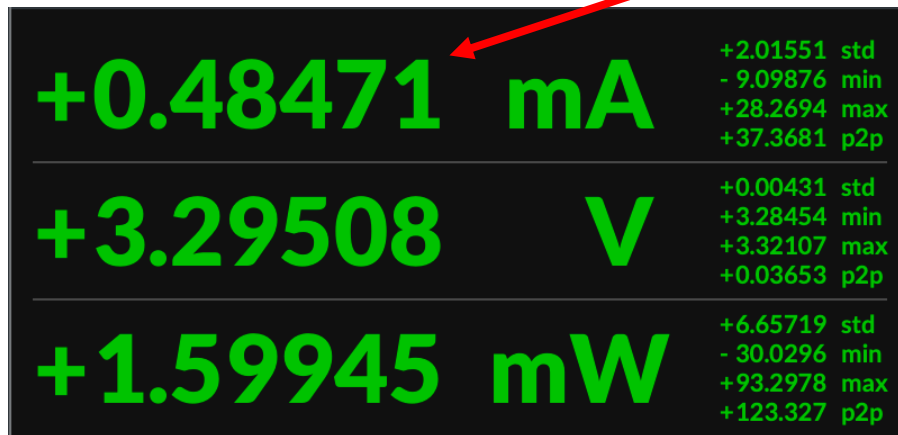
Maybe all pins aren't high-impedance?

Not sure...

And then there's Deep Sleep

```
esp_deep_sleep_start();
```

Mode	Description	Typ (μA)
Light-sleep	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance	130
Deep-sleep	RTC timer + RTC memory	5
Power off	CHIP_EN is set to low level, the chip is powered off	1



Definitely Lower Power but is it good enough?

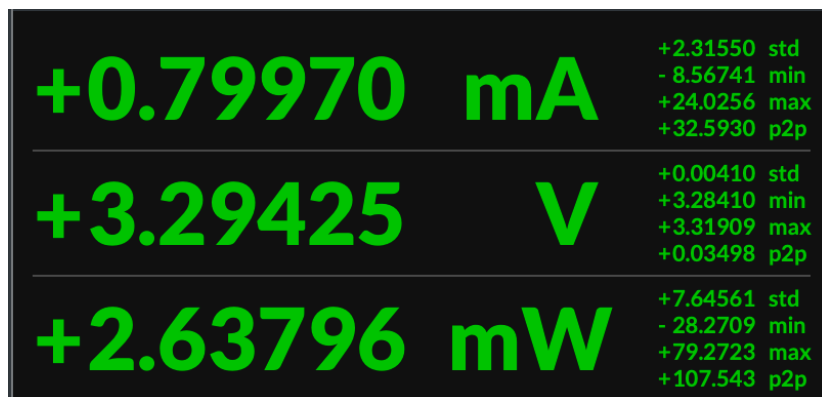
Why is it 100 ish times larger?

Should we worry?

Still...Something Not Right

- The numbers are qualitatively correct when looked at together, but quantitatively in disagreement with the datasheet!

Mode	Description	Typ (μA)
Light-sleep	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance	130
Deep-sleep	RTC timer + RTC memory	5
Power off	CHIP_EN is set to low level, the chip is powered off	1

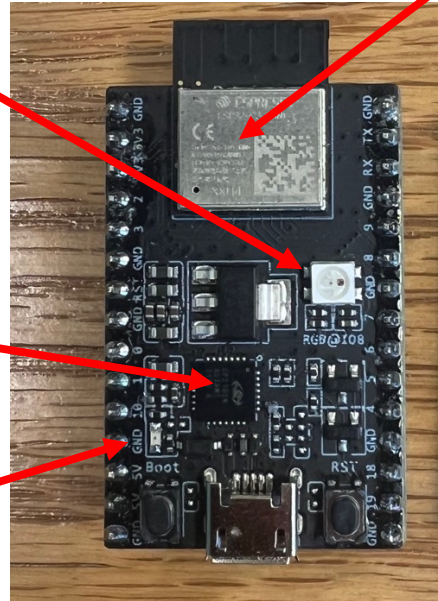


Remember Board

LED that
Comes on
Sometimes

CP2102
USB-UART
Chip

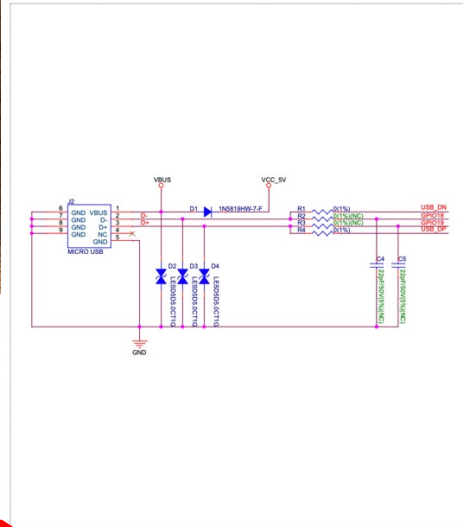
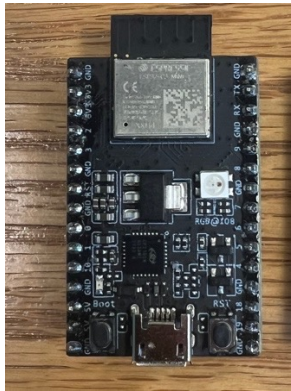
Always-
On
LED



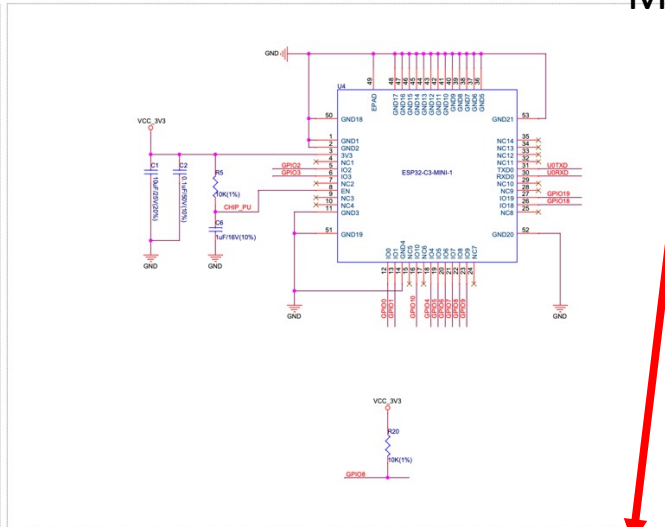
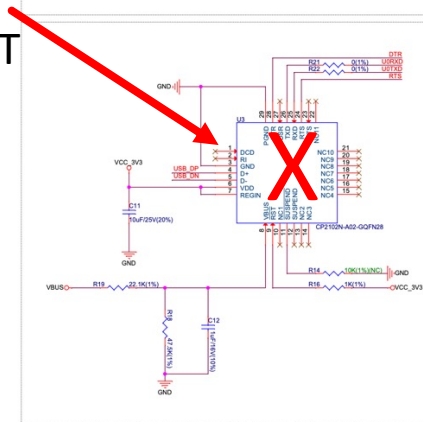
The actual ESP32
C3 is a chip inside
this metal thing

Remember the board? These numbers might be getting impacted by these other rando parts....what are they?

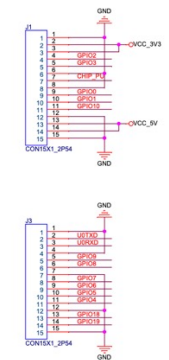
ESP32 C3 Devkit M1 Schematic



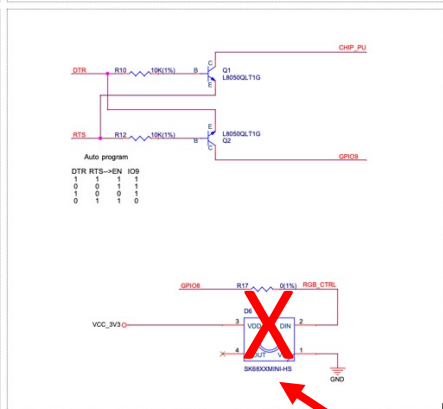
CP2102
USB-UART
Chip



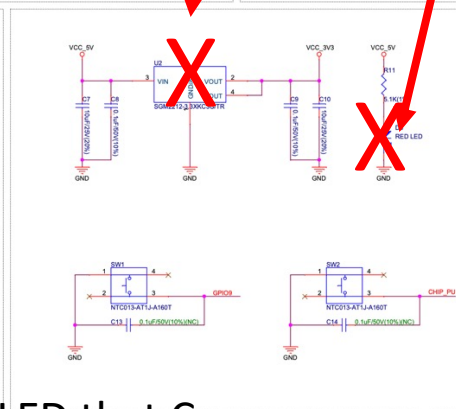
Regulator
Maybe?



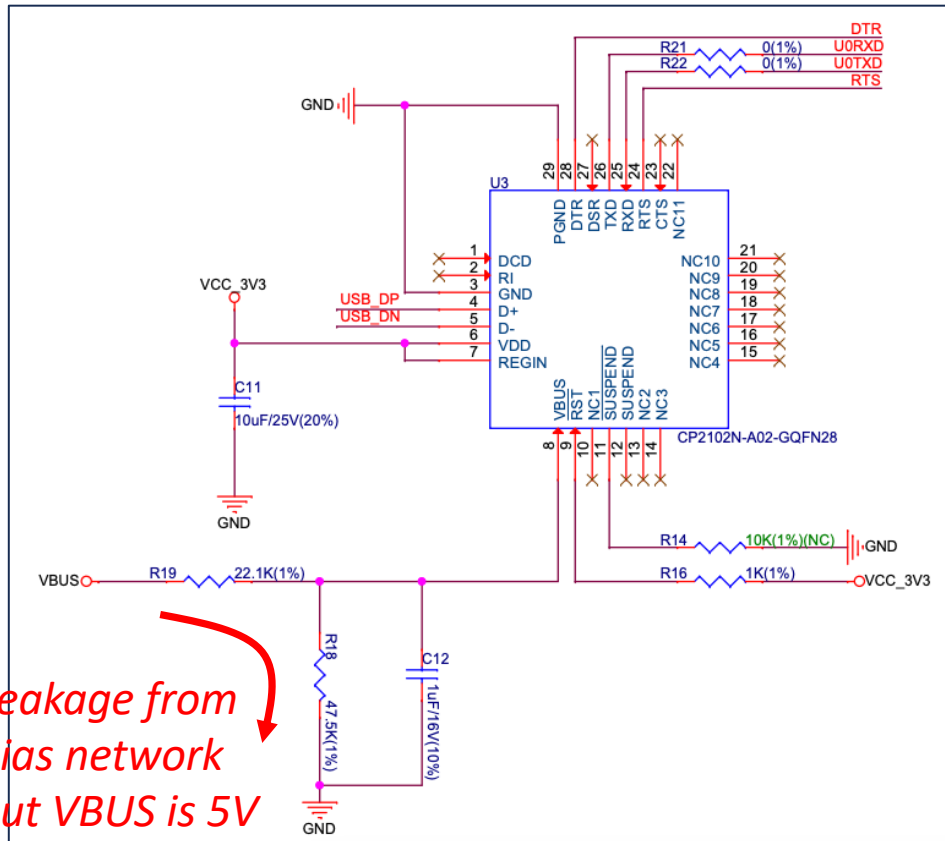
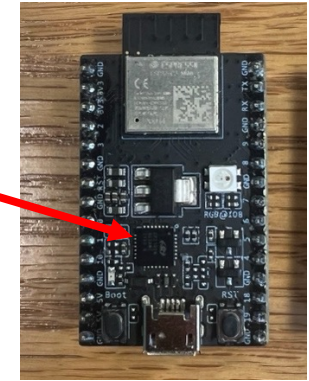
Always-On
LED only
works
with 5V



LED that Comes on
Sometimes and
blinds you



CP2102 USB-UART Interface

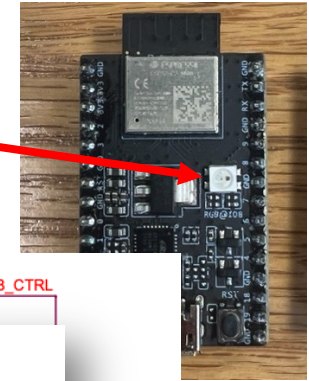


Leakage from bias network but VBUS is 5V and we're not giving board that so ignore

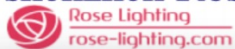
So maybe about 80 uA from that chip just sitting there and doing nothing useful

CP2102						
Supply Current—Normal ³	I_{REGIN}	Normal Operation; V_{REG} Enabled	—	20	26	mA
Supply Current—Suspended ³		Bus Powered; V_{REG} Enabled	—	80	100	μ A

The Bright LED: SK68XXMINI



shenzhen Rose Lighting Technology Co., Ltd



<https://www.rose-lighting.com/> www.ulstrip.com

GPIO8 R17 0(1%) RGB_CTRL

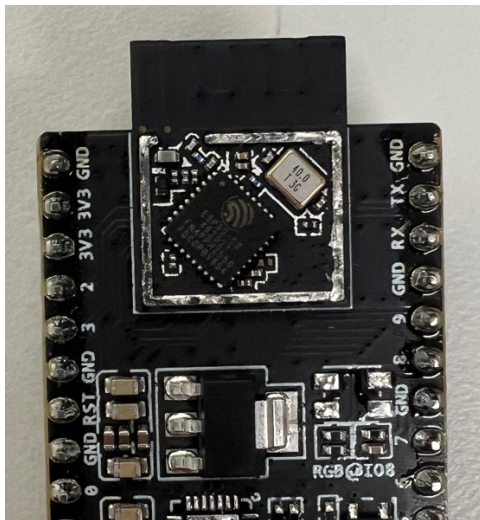
3. Description:

- Top SMD internal integrated high quality external control line serial cascade constant current IC;
- control circuit and the RGB chip in SMD 3535 components, to form a complete control of pixel, color mixing uniformity and consistency;
- built-in data shaping circuit, a pixel signal is received after wave shaping and output waveform distortion will not guarantee a line;
- The built-in power on reset and reset circuit, the power does not work;
- gray level adjusting circuit (256 level gray scale adjustable);
- red drive special treatment, color balance;
- line data transmission;
- plastic forward strengthening technology, the transmission distance between two points over 10M;
- Using a typical data transmission frequency of 800 Kbps, when the refresh rate of 30 frames per sec

The chip supply voltage	VDD	---	5.2	---	V	---
The signal input flip threshold	VIH	0.7*VDD	---	---	V	VDD=5.0V
	VIL	---	---	0.3*VDD	V	
The frequency of PWM	FPWM	---	1.2	---	KHZ	---
Static power consumption	IDD	---	1	---	mA	---

Is there anything else in the Mini Module?

Removed top off of Mini Module:



This is the reference design of the module.

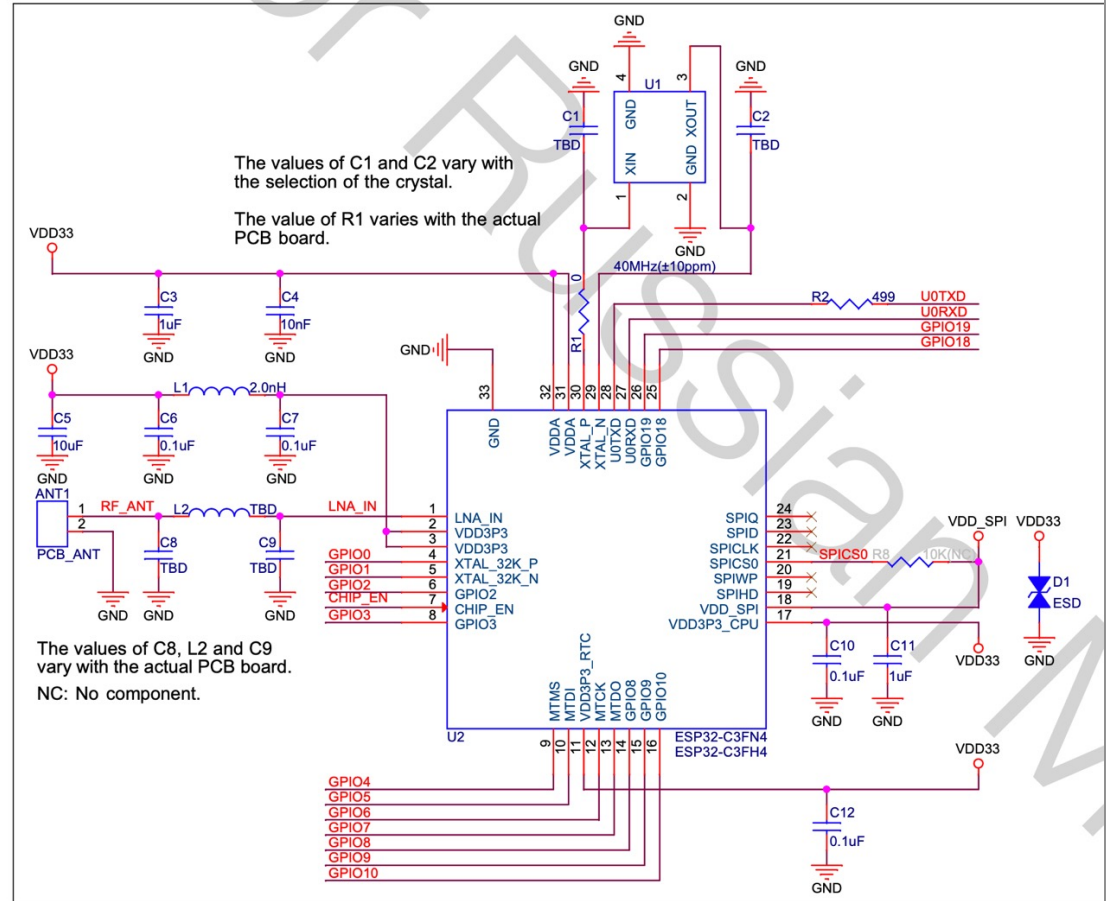


Figure 5: ESP32-C3-MINI-1 Schematics

https://www.espressif.com/sites/default/files/russianDocumentation/esp32-c3-mini-1_datasheet_en.pdf

Is there anything else in the Mini Module?

Nothing...

Ugh....

This is the reference design of the module.

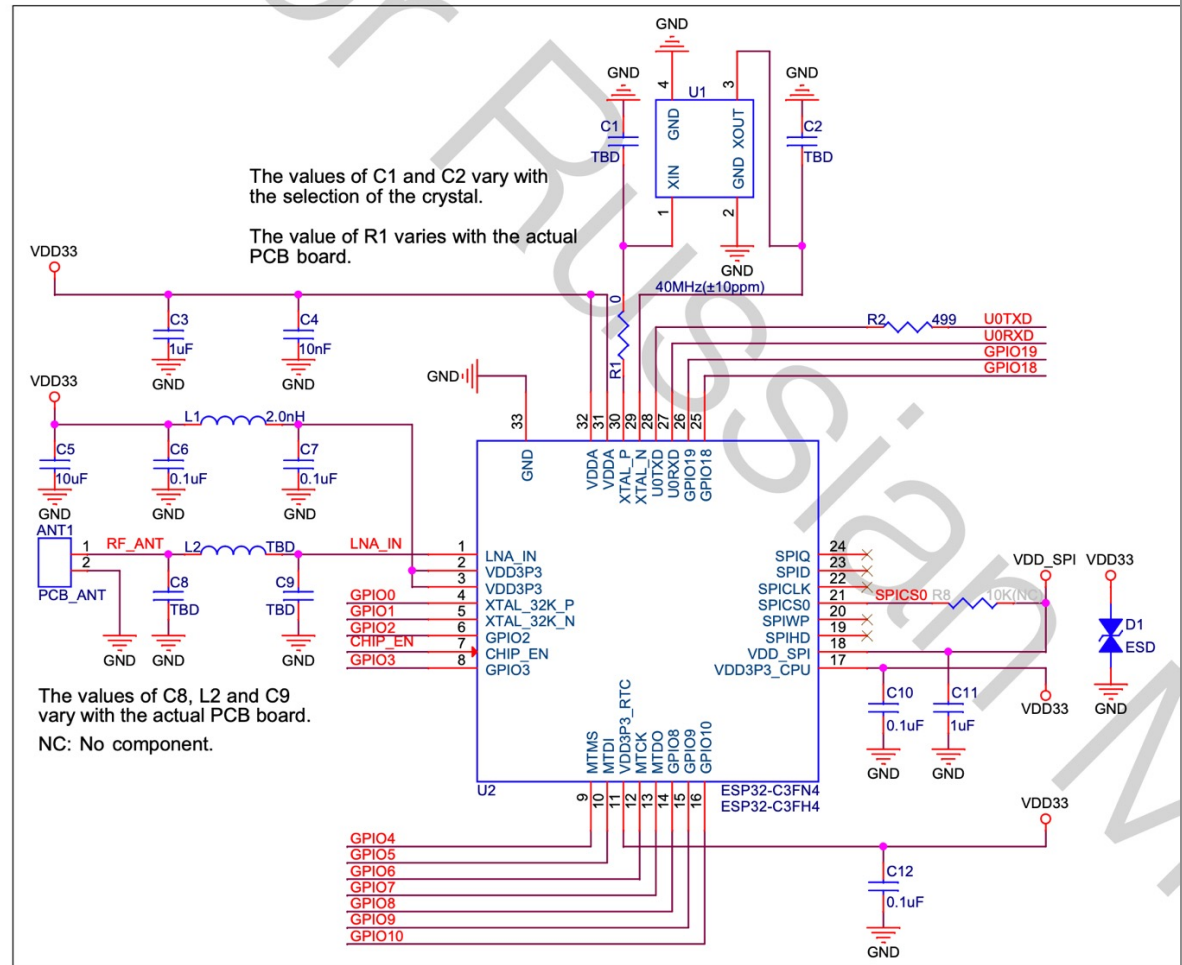
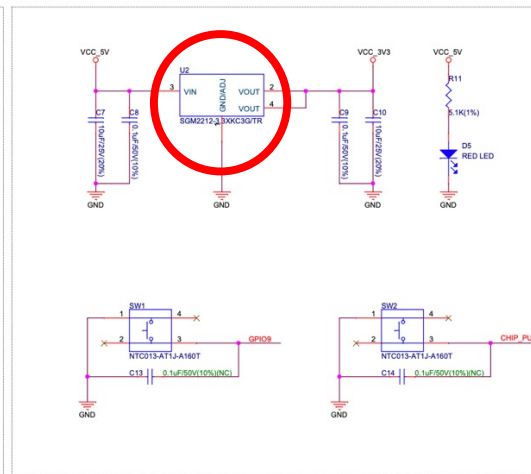
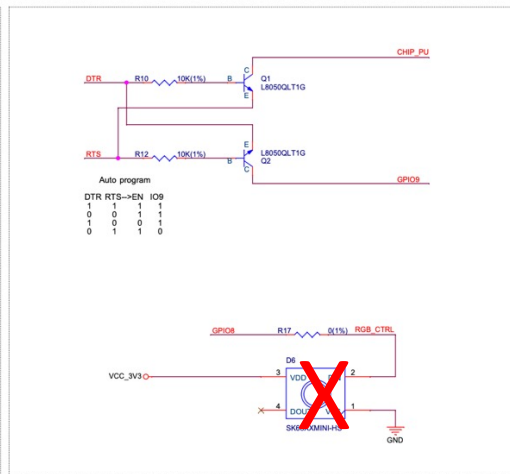
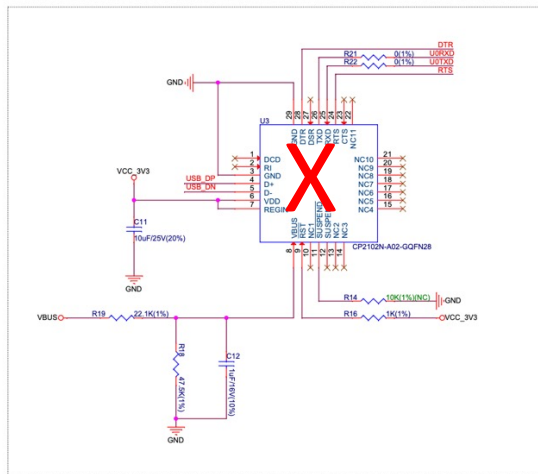
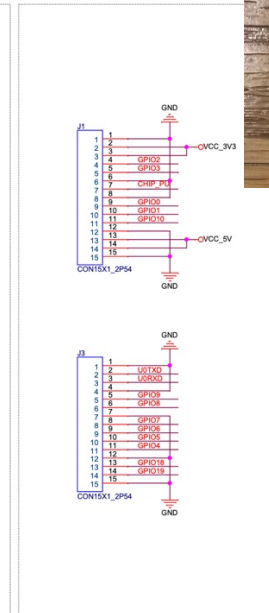
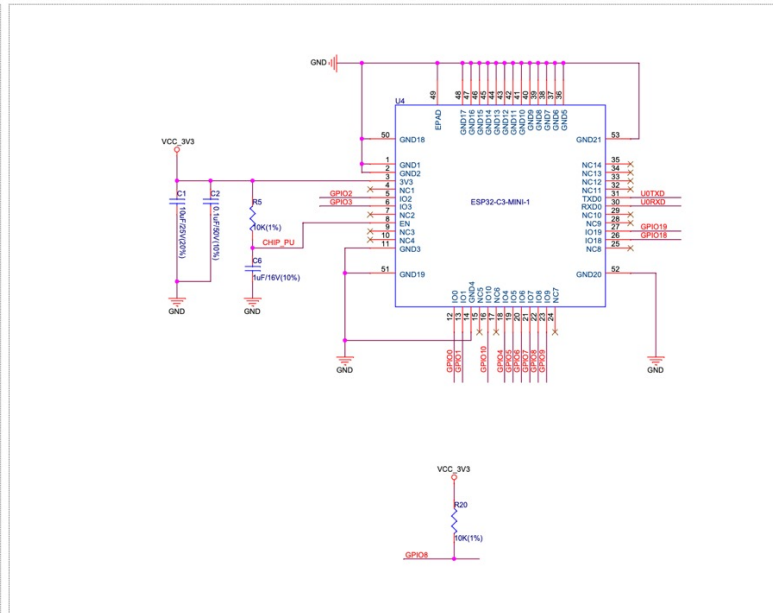
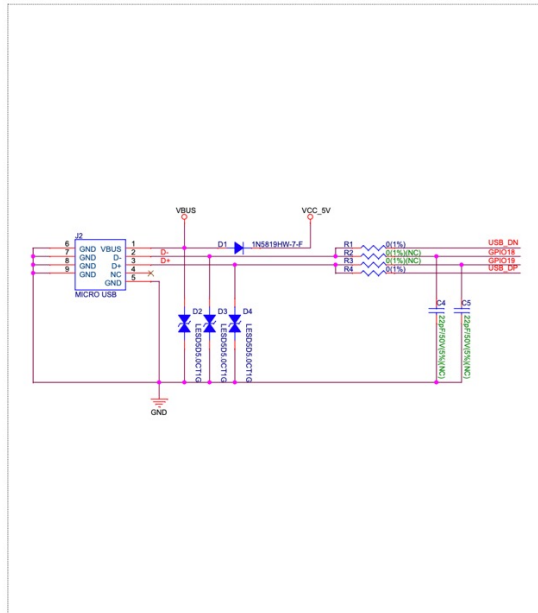
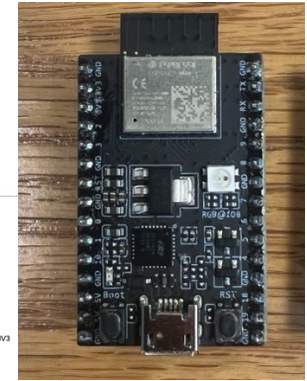


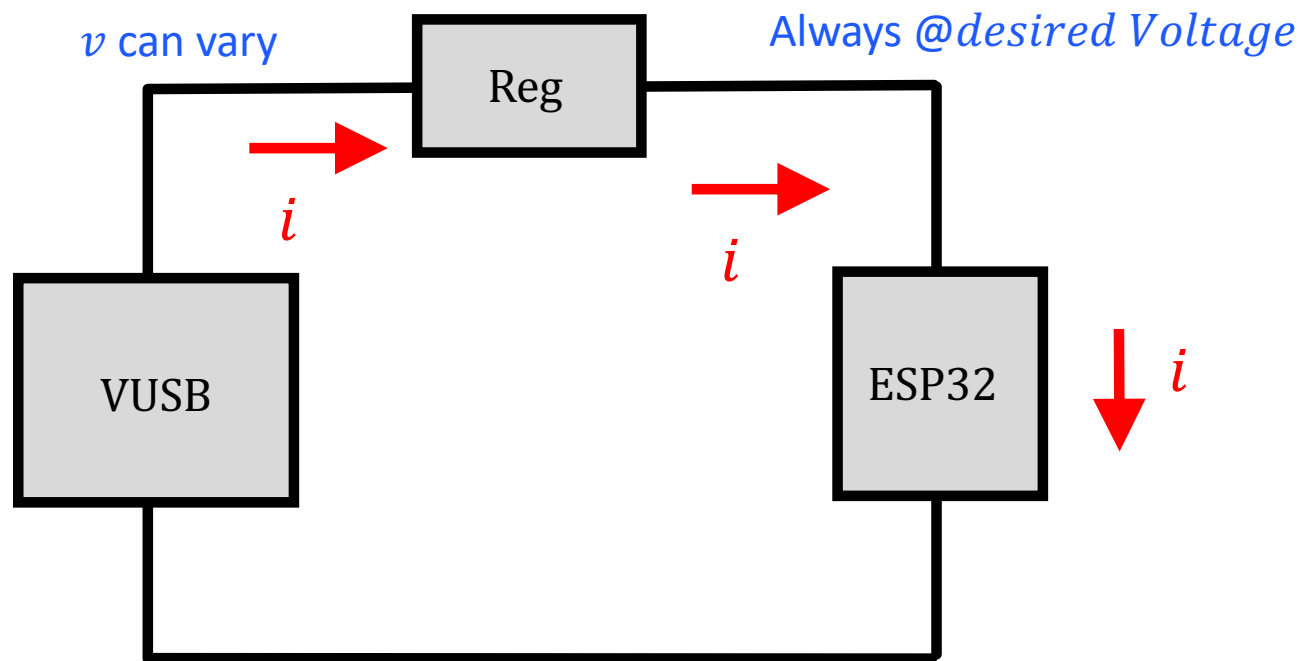
Figure 5: ESP32-C3-MINI-1 Schematics

Back to the Whole Board



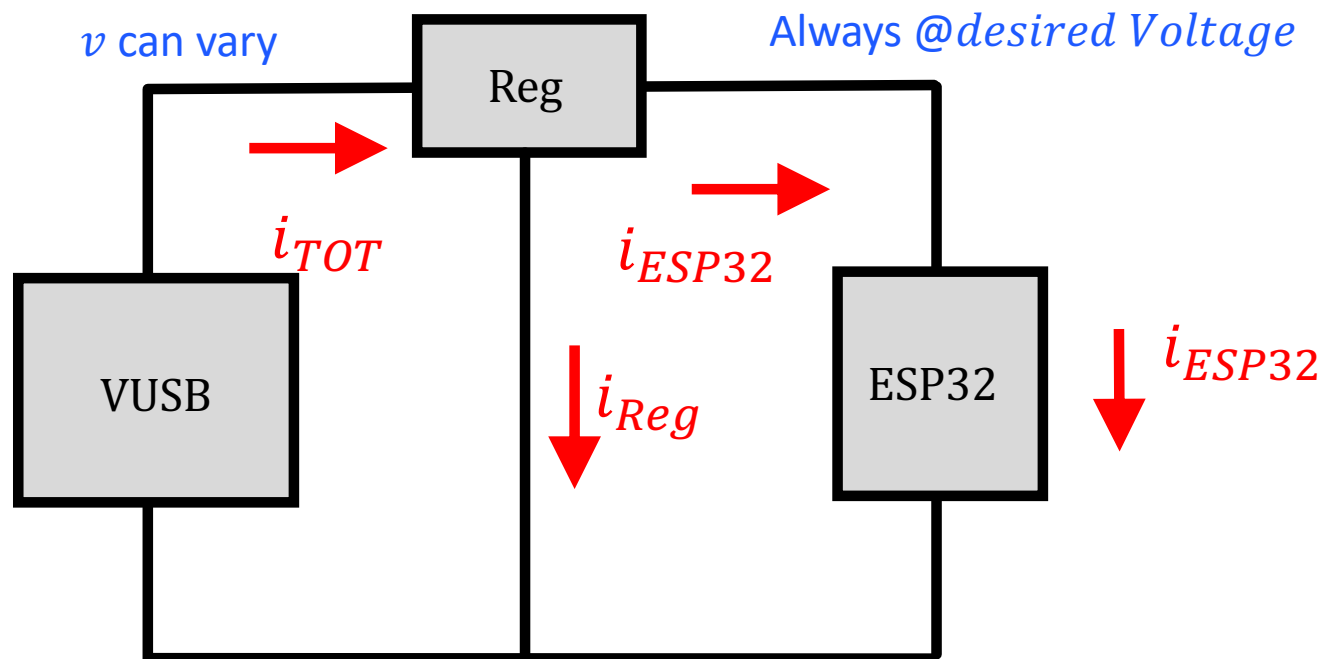
We saw a Linear Regulator Last Week

- Constant Current Device (KCL maintained)
- Can only regulate down in voltage



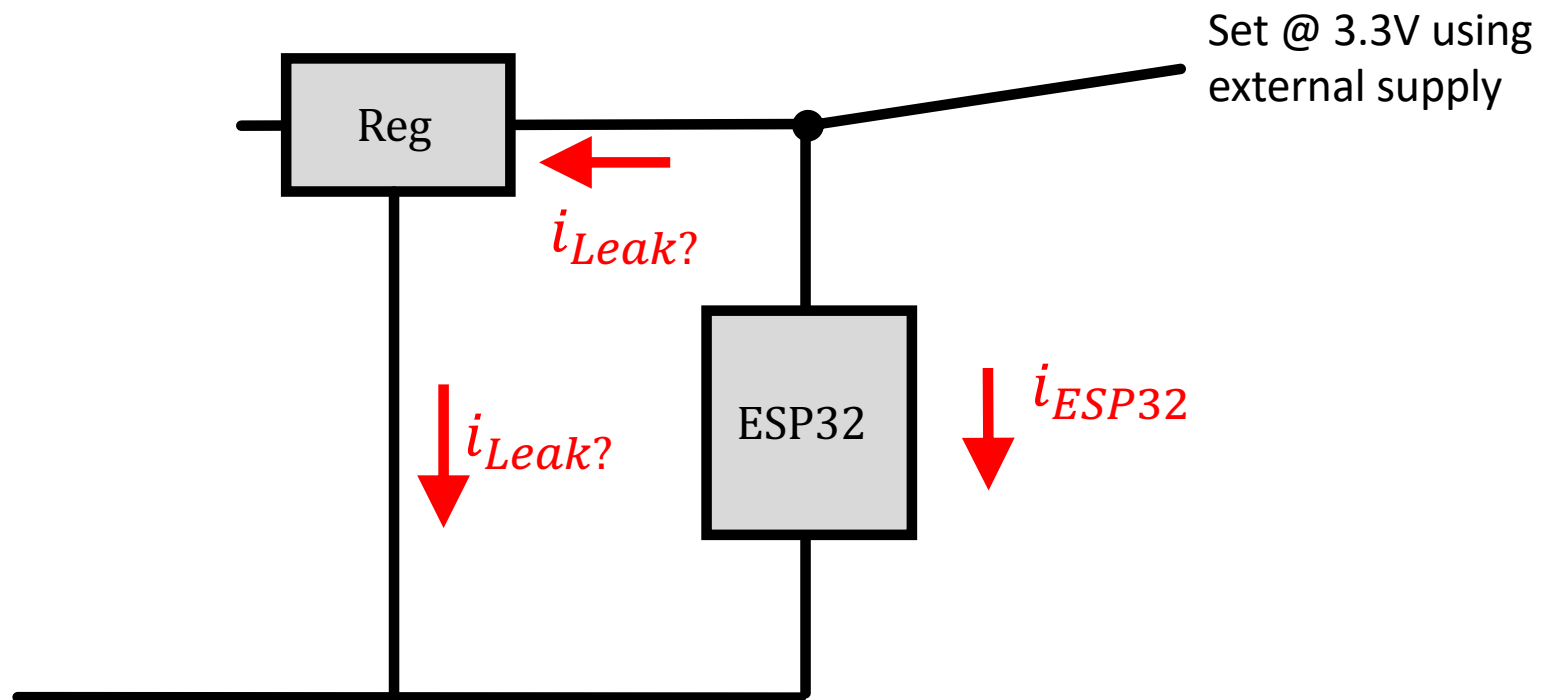
We saw a Linear Regulator Last Week

- Constant Current Device (KCL maintained)
- Can only regulate down in voltage

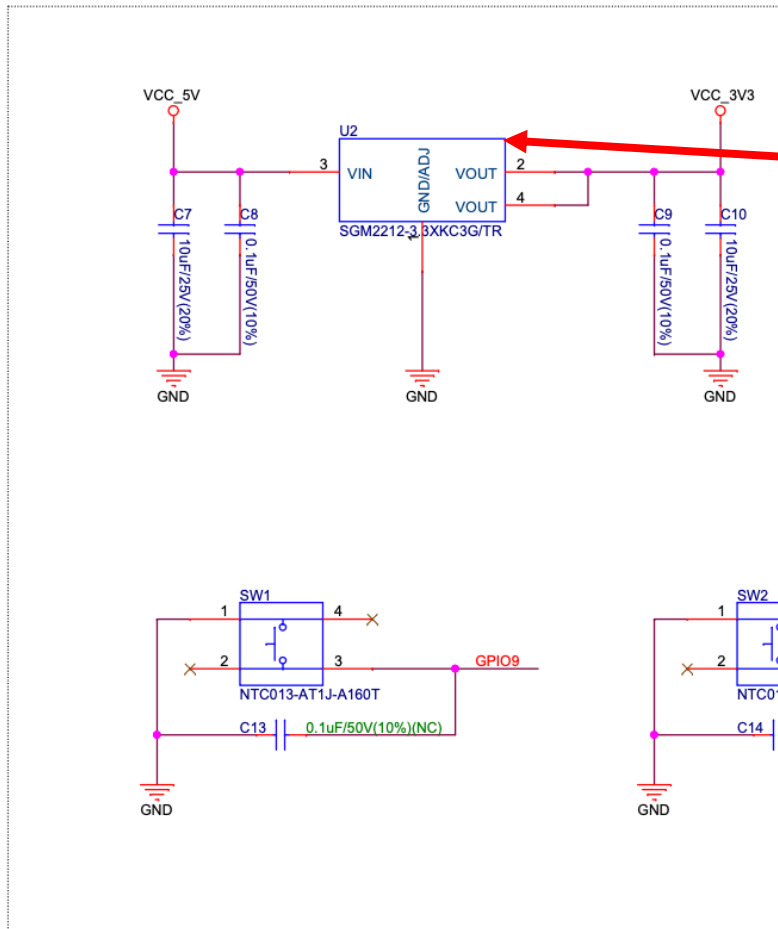


We saw a Linear Regulator Last Week

- No idea how the regulator will work when not actively powered but it



Any particular spots?



Even though the Linear regulator is driven by 5V and there is no 5V in our system as tested, the regulator might be leaking some current from its VOUT terminal to GND

SGM2212



SGM2212 800mA, Low Noise, Low Quiescent Current, High PSRR, Low Dropout Linear Regulator

GENERAL DESCRIPTION

The SGM2212 is a low noise, low quiescent current, high PSRR, fast transient response and low dropout voltage linear regulator which is designed using CMOS technology. It provides 800mA output current capability. The operating input voltage range is from 2.7V to 20V. The fixed output voltages are 1.8V, 2.5V, 2.8V, 3.3V, 5V and adjustable output voltage range is from 1.25V to 15V.

Other features include short-circuit current limit and thermal shutdown protection.

The SGM2212 is available in Green TO-252-2, TO-263-3, SOT-223-3 and TDFN-3x3-8L packages. It operates over an operating temperature range of -40°C to +125°C.

FEATURES

- **Input Voltage Range:** 2.7V to 20V
- **Output Current:** 800mA
- **Fixed Output Voltages:** 1.8V, 2.5V, 2.8V, 3.3V, 5V
- **Adjustable Output Voltage Range:** 1.25V to 15V
- **Output Voltage Accuracy:** ±1% at +25°C
- **Line Regulation:** 0.012% (MAX)
- **Load Regulation:** 0.4% (MAX)
- **Stable with Small Case Size Ceramic Capacitors**
- **Output Current Limit**
- **Thermal Shutdown Protection**
- **-40°C to +125°C Operating Temperature Range**
- **Available in Green TO-252-2, TO-263-3, SOT-223-3 and TDFN-3x3-8L Packages**

APPLICATIONS

Portable Electronic Device

When powered with 5V but no draw on the 3.3V about 80 uA consumed, but no indication how this regulator works when not powered by 5V, but VOUT connected to 3.3V (kinda out of spec)

APPLICATION INFORMATION

The SGM2212 is a low noise, fast transient response high performance LDO, it consumes only 80µA (TYP) quiescent current and provides 800mA output current. The SGM2212 provides the protection function for output overload, output short-circuit condition and overheating.

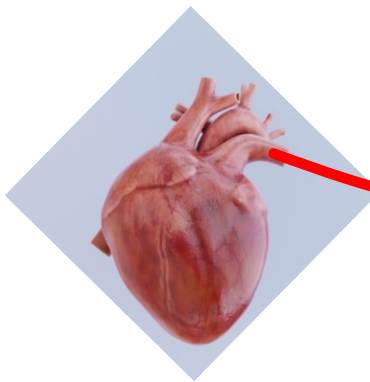
The SGM2212 is suitable for application which has noise sensitive circuit such as battery-powered equipment and smartphones.



<https://www.sg-micro.com/uploads/soft/20220506/1651829970.pdf>

Conclusion: Those Components are Leeches

- They can serve a purpose on the development board for sure, but in this context they are leeches sucking away our precious natural bodily fluids
- Must Purge Them

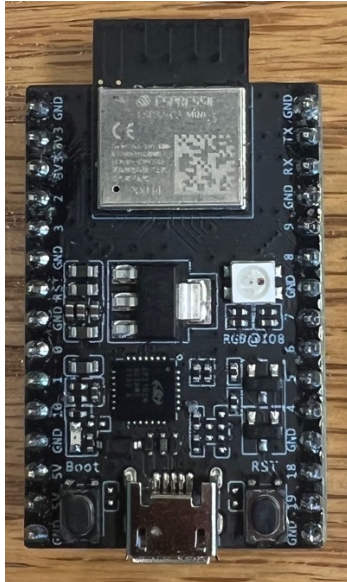


18650 Lithium Battery

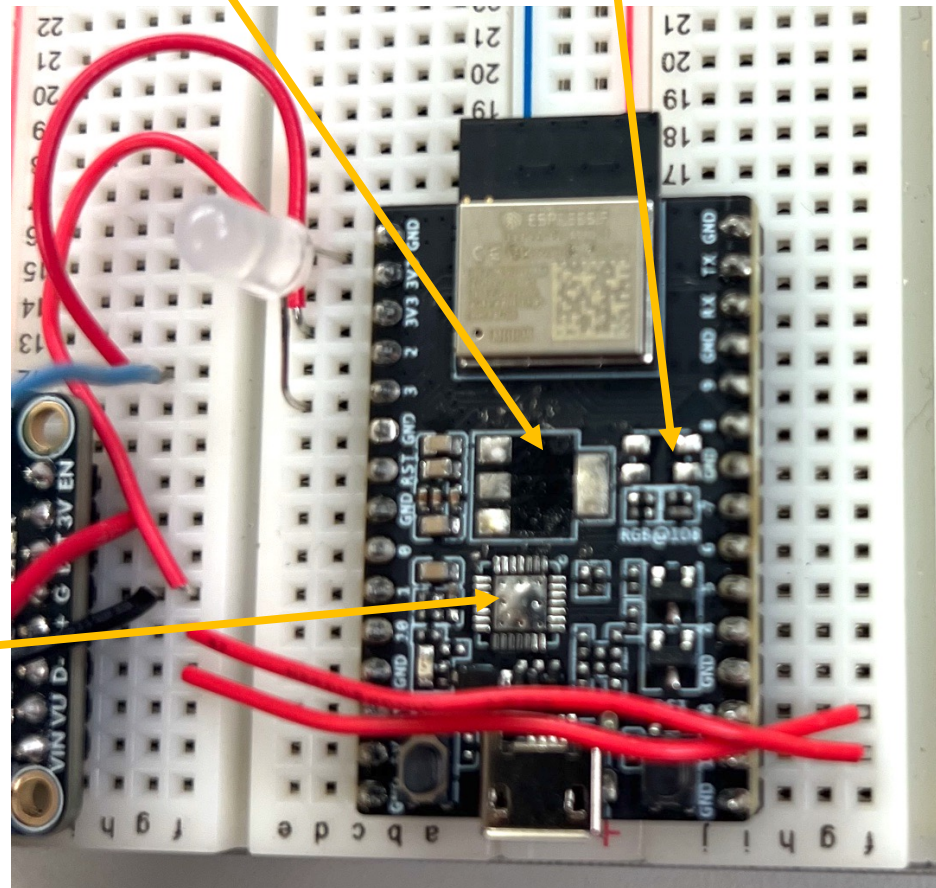


Purge them

Original:



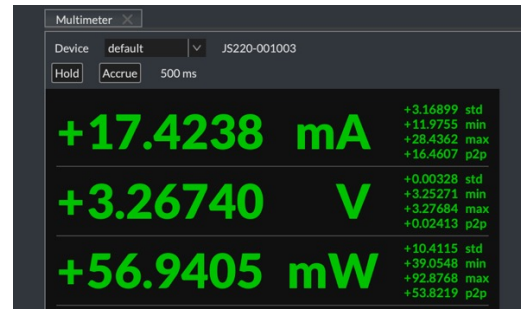
Removed SGM2212 Removed SK68XXMINI



Removed CP2102

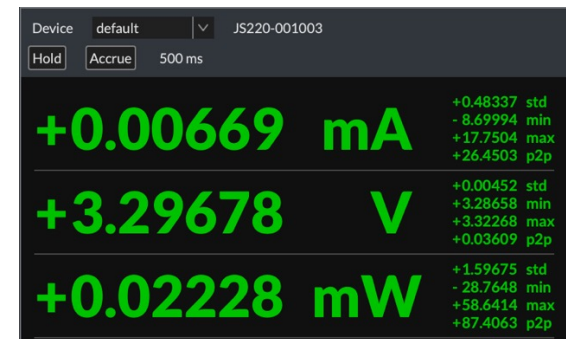
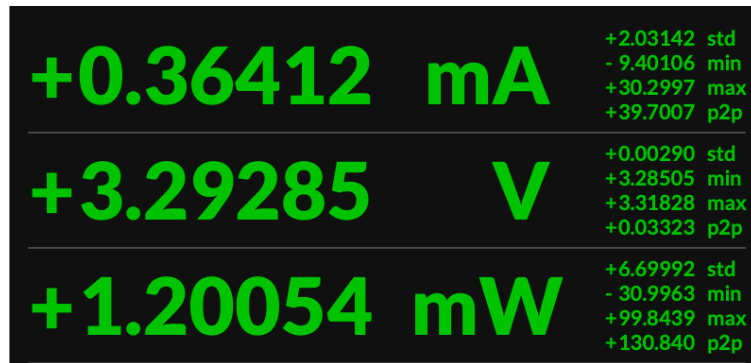
Remove them

Regular Running:



`esp_light_sleep_start();`

`esp_deep_sleep_start();`



Mode	Description	Typ (μ A)
Light-sleep	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance	130
Deep-sleep	RTC timer + RTC memory	5
Power off	CHIP_EN is set to low level, the chip is powered off	1

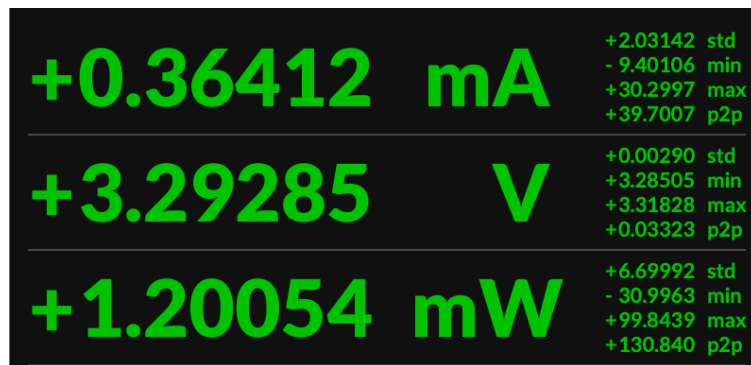
Deep Sleep Actually Gets Down There!!!



So Getting Somewhere!!!

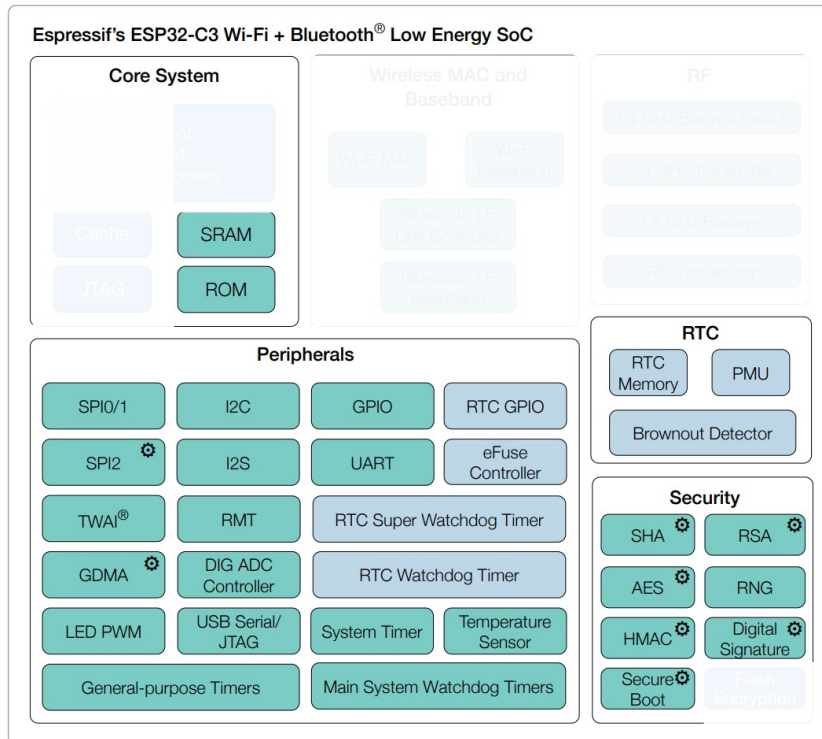
- Deep Sleep seems to actually be working in spec!
- Light sleep...hmmm little bit weird

```
esp_light_sleep_start();
```



Mode	Description	Typ (μA)
Light-sleep	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance	130
Deep-sleep	RTC timer + RTC memory	5
Power off	CHIP_EN is set to low level, the chip is powered off	1

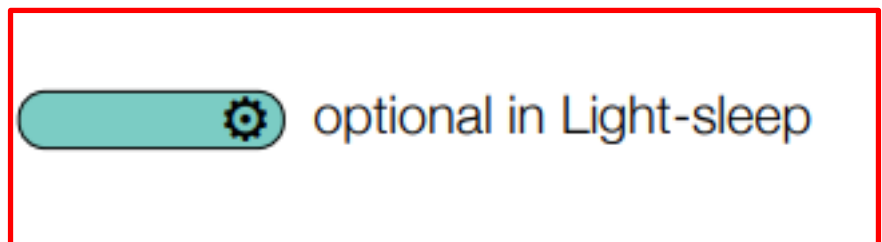
Working Theory is Some of these optional things are still on



Modules having power in specific power modes:

- Active
- Active and Modem-sleep
- Active, Modem-sleep, and Light-sleep; optional in Light-sleep
- All modes

- Everything listed so far
- The processor is off
- Cache is off
- JTAG is off
- Some stuff on but maybe you can turn off

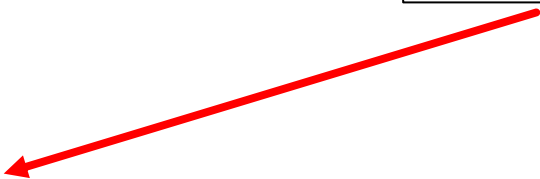


Doubled-checked and ran without Arduino stuff:

```
#include<stdio.h>
#include "esp_sleep.h"
// remove Arduino caca

void app_main() {
    esp_sleep_enable_timer_wakeup(6000000); // 6 sec
    while (1){
        printf("hi there\n");
        esp_deep_sleep_start();
    }
}
```

Goes to this number consistently during light sleep



+0.36412 mA	+2.03142 std - 9.40106 min +30.2997 max +39.7007 p2p
+3.29285 V	+0.00290 std +3.28505 min +3.31828 max +0.03323 p2p
+1.20054 mW	+6.69992 std - 30.9963 min +99.8439 max +130.840 p2p

Power consumption in light sleep is the same...so I don't know

Light Sleep?

```
esp_light_sleep_start();
```

- Also was running another test script where it:
 - Got number fact from numbersapi.com
 - Went to light sleep
- After a certain number of cycles...power consumption got weird

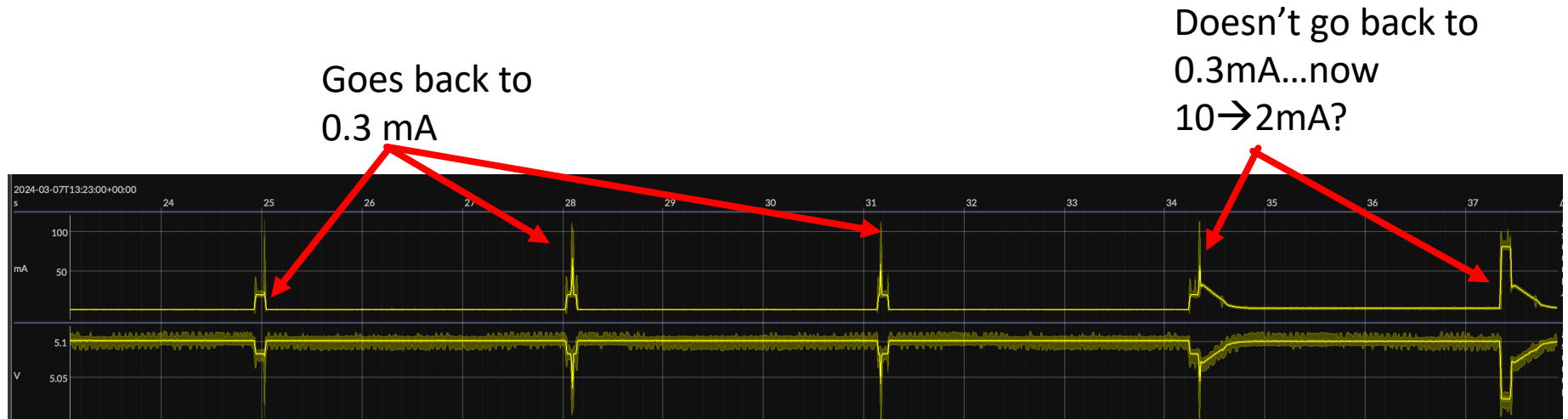
But sometimes in light sleep...



Mode	Description	Typ (μ A)
Light-sleep	VDD_SPI and Wi-Fi are powered down, and all GPIOs are high-impedance	130
Deep-sleep	RTC timer + RTC memory	5
Power off	CHIP_EN is set to low level, the chip is powered off	1

WTF?

During the Light-Sleep-Wake Cycle with periodic Wifi Grabs:



- This needs to be figured out. An inconsistent out-of-spec performance makes more me worried than a consistent out-of-spec performance
- I think it is caused because I didn't turn off Wifi before going to light sleep with a command and they say you're supposed to....but you should figure out.

Must dig through docs and app notes!

ESPRESSIF

ESP32-C3

stable (v5.2.1)

Search docs

Get Started

- API Reference
 - API Conventions
 - Application Protocols
 - Bluetooth® API
 - Error Codes Reference
 - Networking APIs
 - Peripherals API
 - Project Configuration
 - Provisioning API
 - Storage API
- System API
 - App Image Format
 - Bootloader Image Format
 - Application Level Tracing
 - Call Function with External Stack
 - Chip Revision
 - Console
 - eFuse Manager
 - Error Code and Helper Functions
 - ESP HTTPS OTA
 - Event Loop Library
 - FreeRTOS Overview
 - FreeRTOS (IDF)
 - FreeRTOS (Supplemental Features)

Power Management

[中文]

Overview

Power management algorithm included in ESP-IDF can adjust the advanced peripheral bus (APB) frequency, CPU frequency, and put the chip into Light-sleep mode to run an application at smallest possible power consumption, given the requirements of application components.

Application components can express their requirements by creating and acquiring power management locks.

For example:

- Driver for a peripheral clocked from APB can request the APB frequency to be set to 80 MHz while the peripheral is used.
- RTOS can request the CPU to run at the highest configured frequency while there are tasks ready to run.
- A peripheral driver may need interrupts to be enabled, which means it has to request disabling Light-sleep.

Since requesting higher APB or CPU frequencies or disabling Light-sleep causes higher current consumption, please keep the usage of power management locks by components to a minimum.

Configuration

Power management can be enabled at compile time, using the option `CONFIG_PM_ENABLE`.

Enabling power management features comes at the cost of increased interrupt latency. Extra latency depends on a number of factors, such as the CPU frequency, single/dual core mode, whether or not frequency switch needs to be done. Minimum extra latency is 0.2 us (when the CPU frequency is 240 MHz and frequency scaling is not enabled). Maximum extra latency is 40 us (when frequency scaling is enabled, and a switch from 40 MHz to 80 MHz is performed on interrupt entry).

Dynamic frequency scaling (DFS) and automatic Light-sleep can be enabled in an application by calling the function `esp_pm_configure()`. Its argument is a structure defining the frequency scaling

ESP32-C3

Wireless Adventure: A Comprehensive Guide to IoT

RISC-V Wi-Fi Bluetooth ESP-IDF ESP RainMaker

ESPRESSIF

<https://www.espressif.com/sites/default/files/documentation/ESP32-C3%20Wireless%20Adventure.pdf>

CHAPTER 12

https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/api-reference/system/power_management.html

Some other sites about ESP family sleep

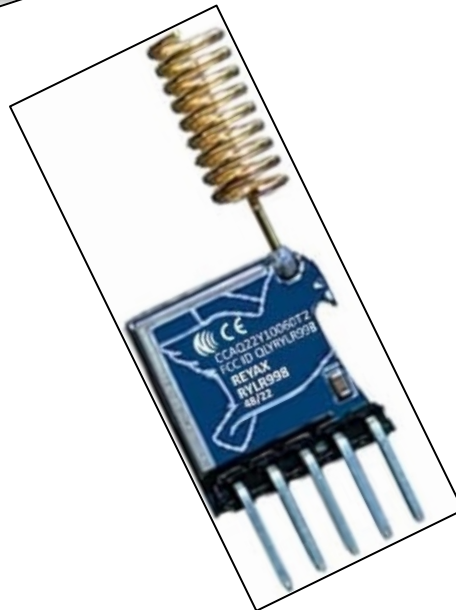
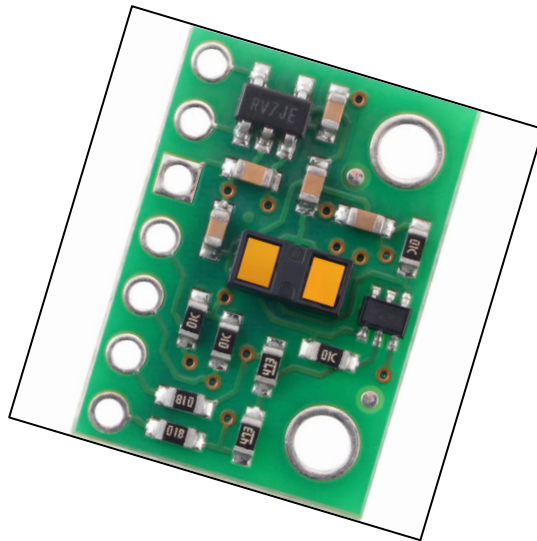
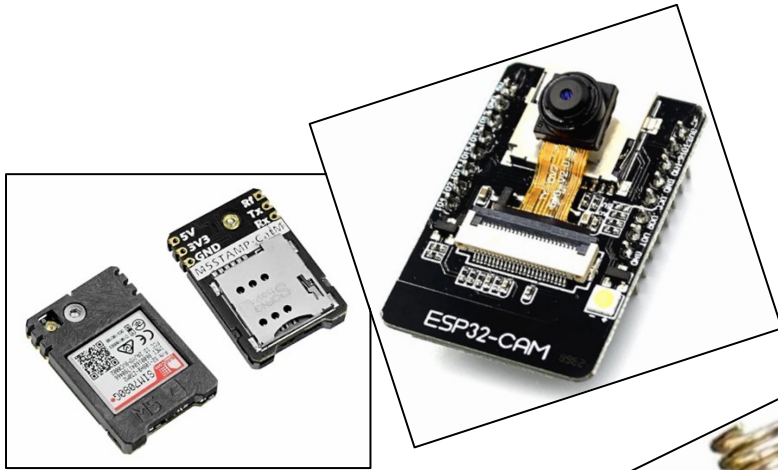
MIGHT HAVE SOME JUICY TIDBITS HERE

<https://blog.voltaicsystems.com/how-to-put-an-esp32-into-deep-sleep/>

<https://randomnerdtutorials.com/esp32-timer-wake-up-deep-sleep/>

Other Parts?

- Each Part that you end up using needs to be vetted just like how our Federal government rigorously vets people they appoint to lead various departments and offices

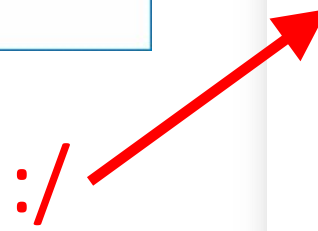


The cellular module from last year was rough...



M5STACK SOLUTION | STORE | SOFTWARE | ABOUT US | DOCUMENTS

Module type	SIM7080G
Support CAT-M band	B1/B2/B3/B4/B5/B8/B12/B13/B14/B18/B19/B20/B25/B26/B27/B28/B66/B85
Support CAT-NB band	B1/B2/B3/B4/B5/B8/B12/B13/B18/B19/B20/B25/B26/B28/B66/B71/B85
Cat-M upstream and downstream speed	Uplink: 1119Kbps Downlink:589Kbps
NB-IoT upstream and downstream speed	Uplink: 150Kbps Downlink:136Kbps
RF Power Class	Class 5 (Typ. 21dbm)
SIM Card Slot Specifications	MicroSIM
Standby Operating Current	DC5V/46mA
Network current	DC5V/71mA
Communication Interface	UART: baud 115200 8N1
Net Weight	4.5g
Gross Weight	17.8g
Product Size	30.1*20.1*5.5mm
Package Size	91.1*135.5mm



<https://shop.m5stack.com/products/m5stamp-cat-m-module-sim7080g>

SIM7080G Module

- The internet and various semi-legitimate sources seemed to suggest there is a sleep mode for this module...
- At that point, it consumes 1.2 mA (not great not terrible)

- Operating Frequency
 - GNSS L1: 1575.42±1.023MHz
 - GLONASS: 1597.5~1605.8 MHz
 - BeiDou: 1559.05~1563.14 MHz
 - Galileo L1: 1575.42±1.023MHz
- Refreshing frequency: 1 Hz (by default)
- GNSS data format: NMEA-0183
- GNSS antenna: active antenna

Other Parameters

- Power supply: 5V
- Logic level: 5V / 3.3V (Switchable via jumper caps)
- Overall current (idle mode): 39mA
- Module sole current (VBAT=3.8V)
- Idle mode: 10mA
- Sleep mode: 1.2mA
- PSM mode: 3.2uA
- eDRX mode: 0.59mA (eDRX=81.92s)
- Operating temperature: -40°C ~ 85°C
- Storage temperature: -45°C ~ 90°C
- Dimension: 30.5mm x 65mm

https://www.waveshare.com/wiki/SIM7080G_Cat-M/NB-IoT_HAT

Maker of Chip explains how to get there

Program it

Might need to use DTR pin??



3.3.2 RI and DTR Behavior

The RI pin description:

The RI pin can be used to interrupt output signal to inform the host controller such as application CPU. Before that, users must use AT command “AT+CFGRI=1” to enable this function.

Normally RI will keep high level until certain conditions such as receiving SMS, or a URC report coming, then it will output a low level pulse 120ms, in the end, it will become high level.

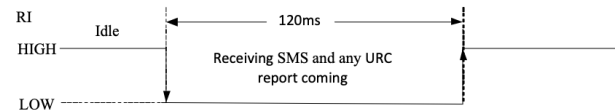


Figure 14: RI behaviour (SMS and URC report)

The DTR pin description:

After setting the AT command “AT+CSCLK=1”, and then pulling up the DTR pin, SIM7080G will enter sleep mode when module is in idle mode. In sleep mode, the UART is unavailable. When SIM7080G enters sleep mode, pulling down DTR can wake up module.

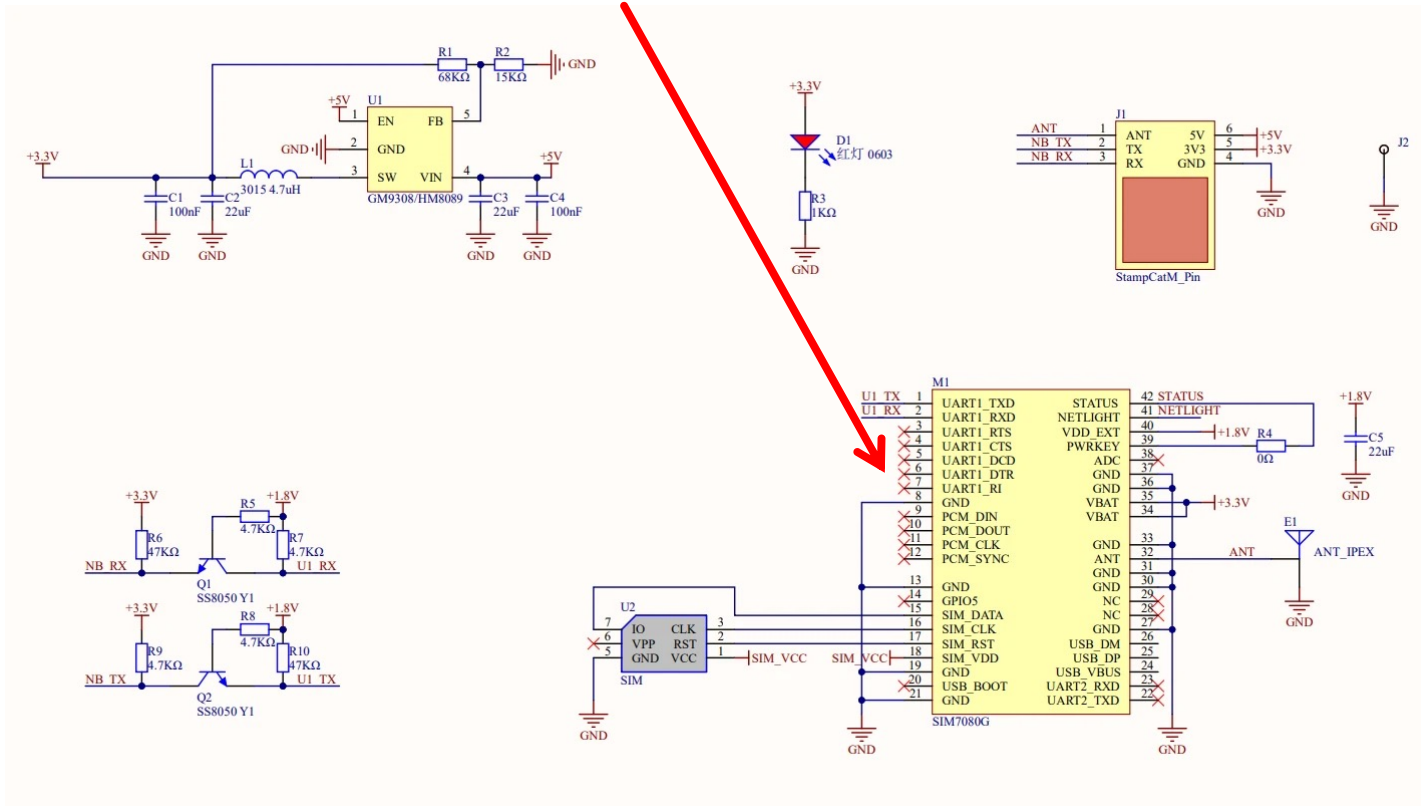
After setting the AT command “AT+CSCLK=0”, SIM7080G will do nothing when the DTR pin is pulling up.

Note: For more details of AT commands about UART, please refer to document [1] and [20].

https://www.waveshare.com/w/upload/e/e7/SIM7080G_Hardware_Design_V1.03.pdf

DTR Pin?

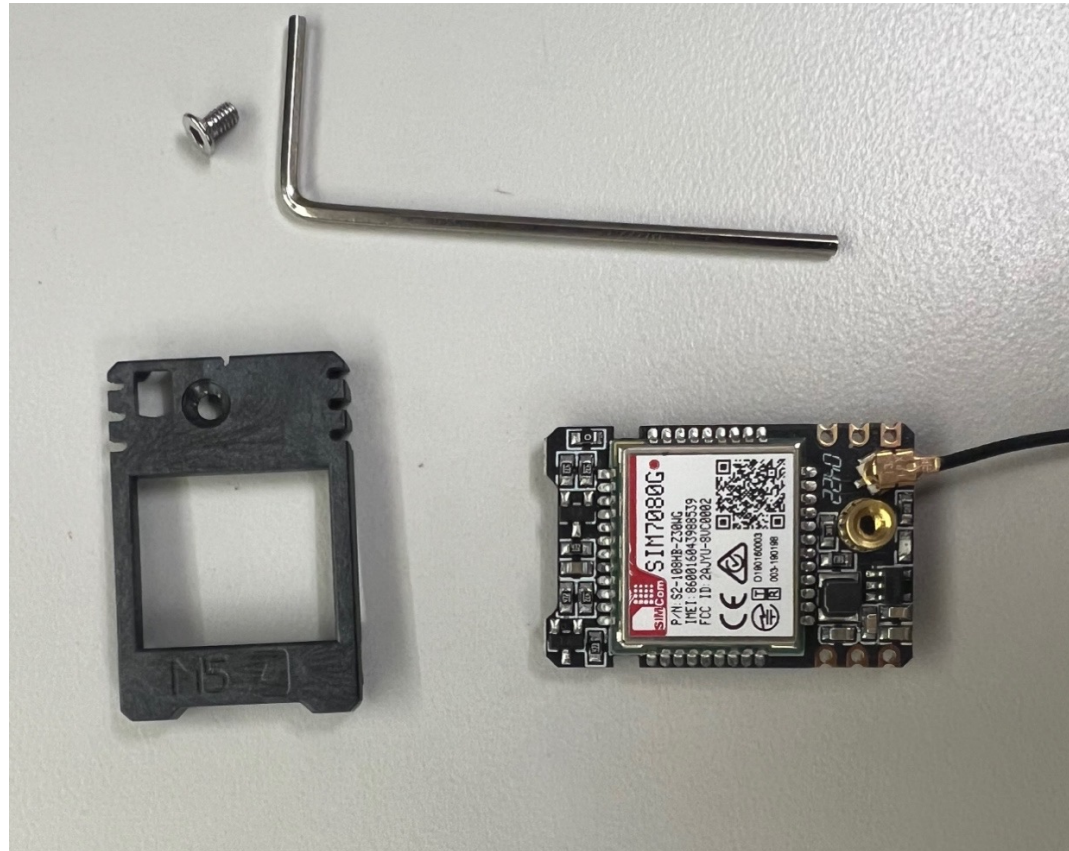
DTR



https://docs.m5stack.com/en/stamp/stamp_catm

Needed to solder a wire onto the dev board thing

- That's ok,
- The pins are pretty accessible
- But this needs to be figured out!



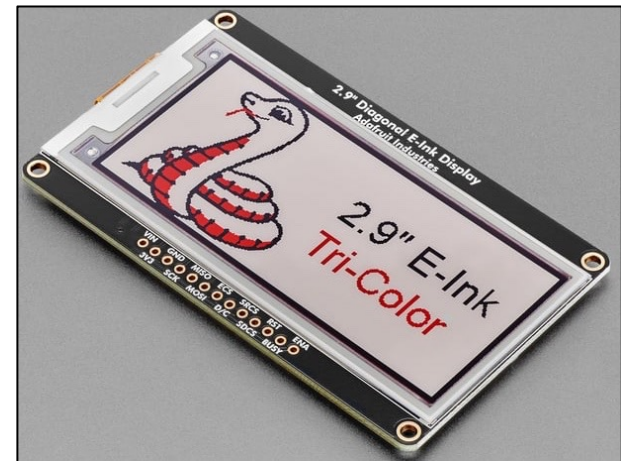
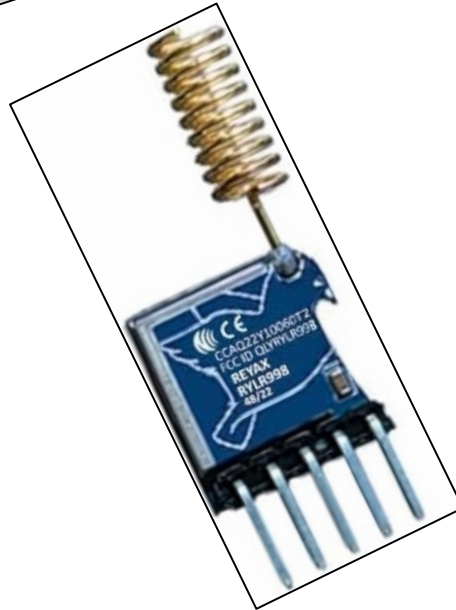
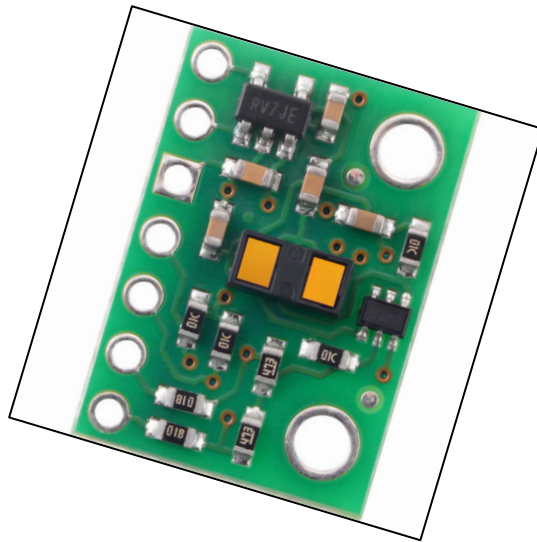
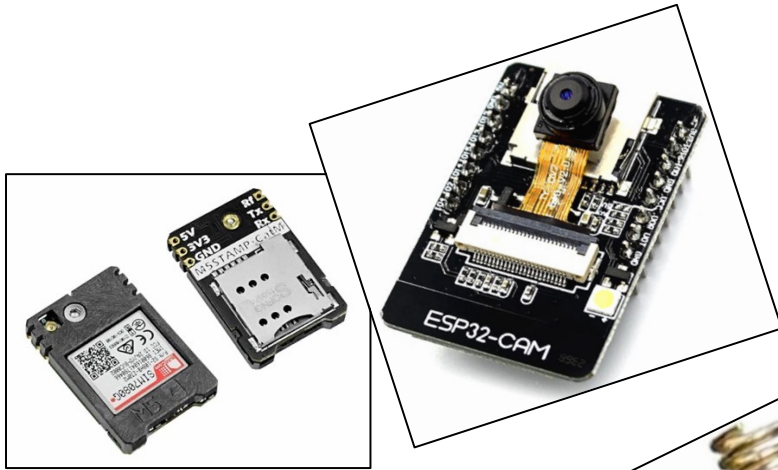
Cell Module



- After all that, I'm actually not even sure if we ever got 1.2 mA out of this thing last year anyways...it was a energy hog.
- Got to plan ahead for this year.

Other Parts?

- Each one is going to be a journey...you will need to do some reverse engineering and thinking about stuff and how it is being used!



Power Budget!

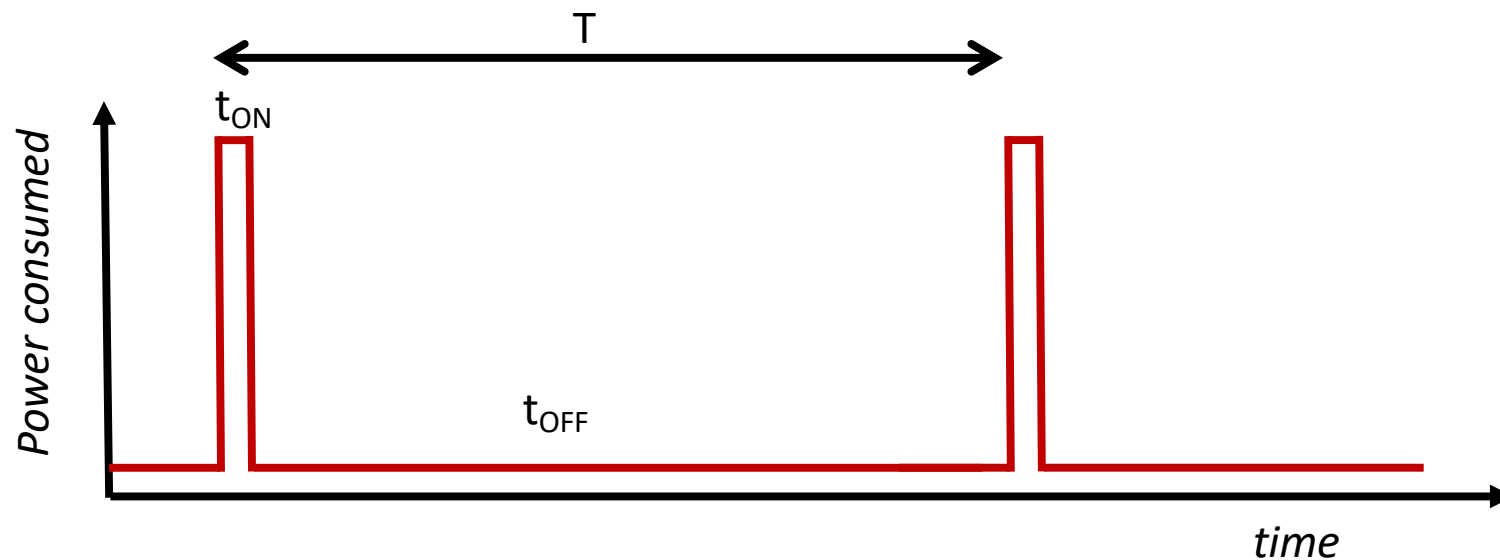
Power Budget

- Now that you're getting an idea of the power of your components you need to engineer with that.
- We'll be relying on placing devices into different power modes in loops whenever possible to make them utilize less energy
- These patterns of power consumption can then be roughly time-averaged and help us determine how much our average consumption will be.
- This can then help us figure out how much we need to be putting into the system (from solar, etc...)
- And how much we need to store up for when we don't have power input (at night for example)

Duty Cycling

- The core of this budgeting will be duty-cycling the components and behavior when possible
- Turn on in bursts, do what is needed, and turn off otherwise.
- The ratio of the “ON” to overall is called the duty cycle

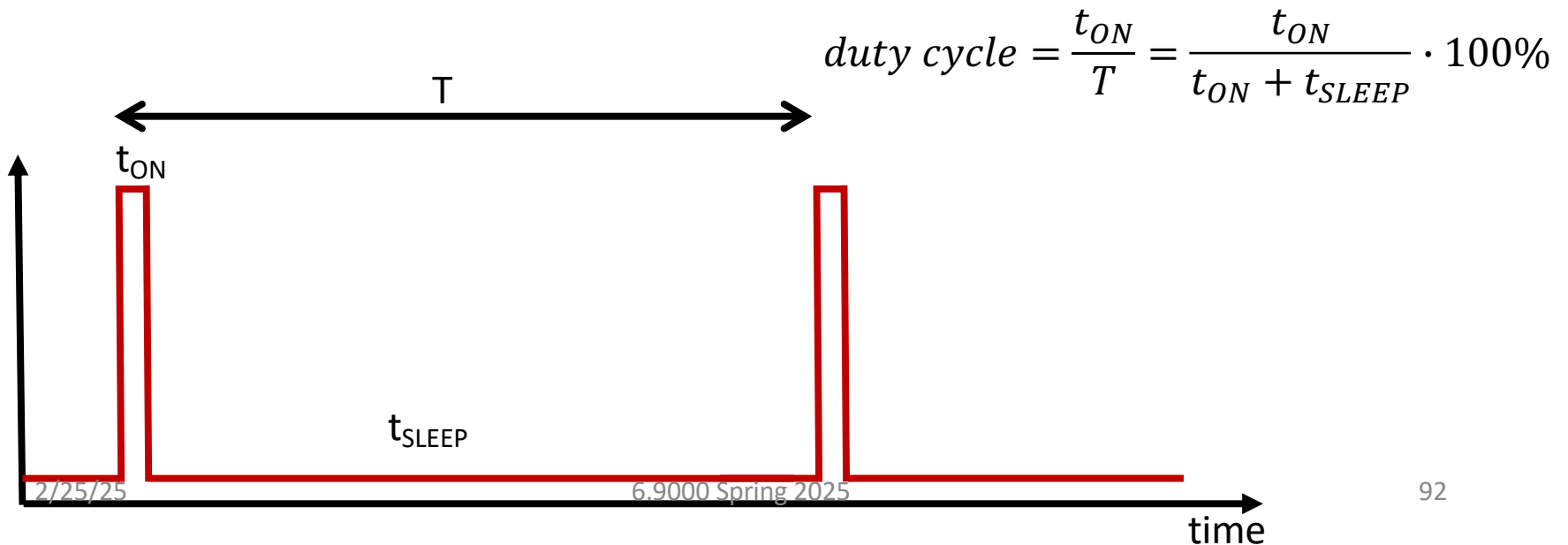
$$\text{duty cycle} = \frac{t_{ON}}{T} = \frac{t_{ON}}{t_{ON} + t_{OFF}} \cdot 100\%$$



Duty Cycling

- Example:

- We have an ESP32 powered with a 200 mAh coin-cell battery. Convert to 3.3V at 100% efficiency
- We want it to run for one month
- We want to take measurements and send them over WiFi, but to measure and connect to WiFi requires a $t_{ON} = 10$ sec.
- How often can we make measurements/send them to the server if we want the coin cell to last for one month?



Duty Cycling

- Adjust Duty Cycle to achieve Average Current:

$$= \frac{t_{on} \cdot i_{on} + t_{sleep} \cdot i_{sleep}}{T} = 0.278 \text{ mA}$$

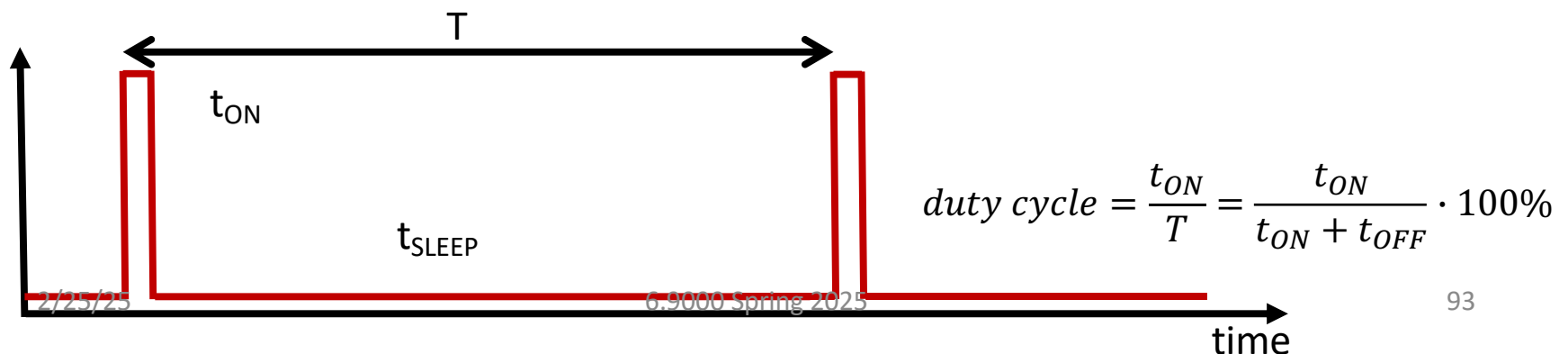
Plug in known values and rewrite in terms of things we're solving for:

$$= \frac{10\text{s} \cdot 150\text{mA} + t_{sleep} \cdot 0.005\text{mA}}{10\text{s} + t_{sleep}} = 0.278 \text{ mA}$$

Solve for t_{sleep} :

$$t_{sleep} \approx 5395\text{s}$$

So we'd need to:
Come on for 10 seconds
Sleep for 5400 seconds
Duty Cycle: 0.185%

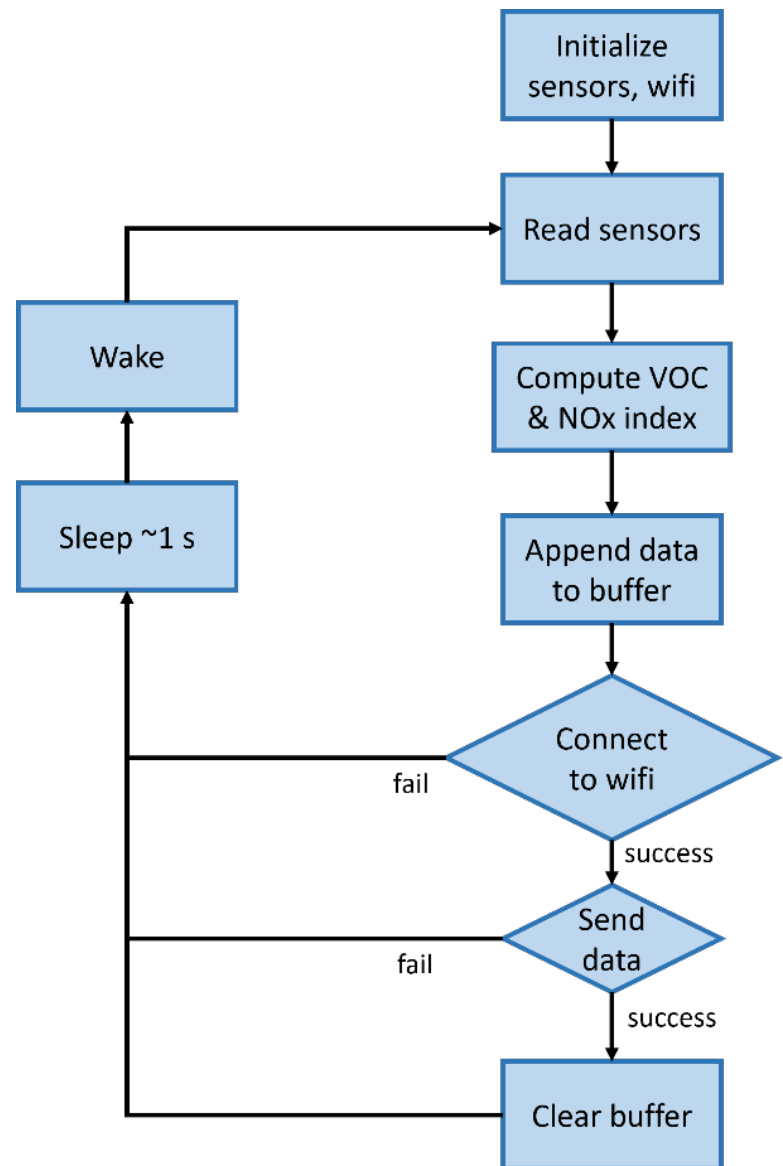


Power Budget

- Now that you're getting an idea of the power of your components you need to engineer with that.
- The first step is to identify your entire system's flowchart of operation.
- Functionality must come first since we need this system to do a thing so you should start with your "ideal" flowchart of operations

Power Budget

- An example from two years ago using the dev boards
- Very similar to some functionality you'll need this year.
- For each state go through and determine what is "on" what is "off" and all the shades of grey in between



Different System States:

- Fill out the budget as you go along

Power budget	Read sensors	Math	Xmit data	Sleep	
MCU subsystem					
ESP32C3	28.00	28.00	345.00	0.13	
LED	5.00	5.00	5.00	5.00	
	33.00	33.00	350.00	5.13	
Sensor subsystem					
SGP41	3.00	0.03	0.03	0.03	
SHTC3	0.90	0.07	0.07	0.07	
LED	5.00	5.00	5.00	5.00	
I2C	0.66	0.33	0.33	0.00	
	9.56	5.43	5.43	5.10	
Power subsystem					
MCP73871	0.03	0.03	0.03	0.03	
AP7361C	0.06	0.06	0.06	0.06	
LEDs	5.00	5.00	5.00	5.00	
Therm	0.05	0.05	0.05	0.05	
PROG1	3.70	3.70	3.70	3.70	
	8.84	8.84	8.84	8.84	
cycle time (ms)	51.40	47.27	364.27	19.07	total current (mA)
10000	70	100	200	9630	duration (ms)
	3.6	4.7	72.9	183.7	charge (mA-sec)
	1.4%	1.8%	27.5%	69.4%	% energy
				Average current/cycle	26.49 mA
				Battery capacity	2200 mA-h
				Lifetime	83.1 h

- Identify problem regions and adjust
- LEDs were always on on these dev boards...

Power budget					
	Read sensors	Math	Xmit data	Sleep	
MCU subsystem					
ESP32C3	28.00	28.00	345.00	0.13	
LED	5.00	0.00	0.00	0.00	
	33.00	28.00	345.00	0.13	
Sensor subsystem					
SGP41	3.00	0.03	0.03	0.03	
SHTC3	0.90	0.07	0.07	0.07	
LED	5.00	0.00	0.00	0.00	
I2C	0.66	0.33	0.33	0.00	
	9.56	0.43	0.43	0.10	
Power subsystem					
MCP73871	0.03	0.03	0.03	0.03	
AP7361C	0.06	0.06	0.06	0.06	
LEDs	5.00	5.00	5.00	5.00	
Therm	0.05	0.05	0.05	0.05	
PROG1	3.70	3.70	3.70	3.70	
	8.84	8.84	8.84	8.84	
cycle time (ms)	51.40	37.27	354.27	9.07	total current (mA)
10000	70	100	200	9630	duration (ms)
	3.6	3.7	70.9	87.4	charge (mA-sec)
	2.2%	2.3%	42.8%	52.8%	% energy
			Average current/cycle	16.56	mA
			Battery capacity	2200	mA-h
			Lifetime	132.9	h

Focus

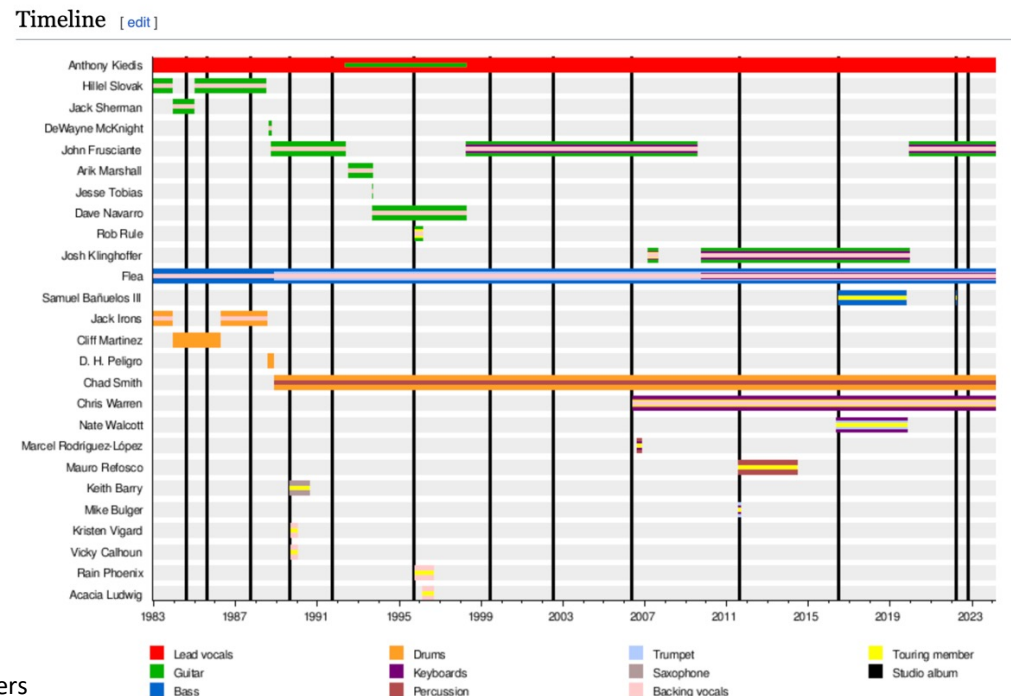
- You need to figure out what your average current draw is going to be!
- **That number must be less than the average that your source can provide**
- And ideally at least several factors smaller than what the source can provide to account for life sucking and never working out how you want

Average current/cycle	16.56	mA		
Battery capacity	2200	mA-h		
Lifetime	132.9	h		

- Lifetime can also give you a sense of how long you can go when it is dark out or cloudy or whatever

Timeline

- When you have many components coming on in various states another way to draw this out might be like what they do to show membership of bands on Wikipedia
- Sort of like a gant chart
- There is free software
- ...to do this



https://en.wikipedia.org/wiki/List_of_Red_Hot_Chili_Peppers_band_members

Timeline

- But we also want to see actual numbers calculated
- And a spreadsheet will likely be the best choice for this.

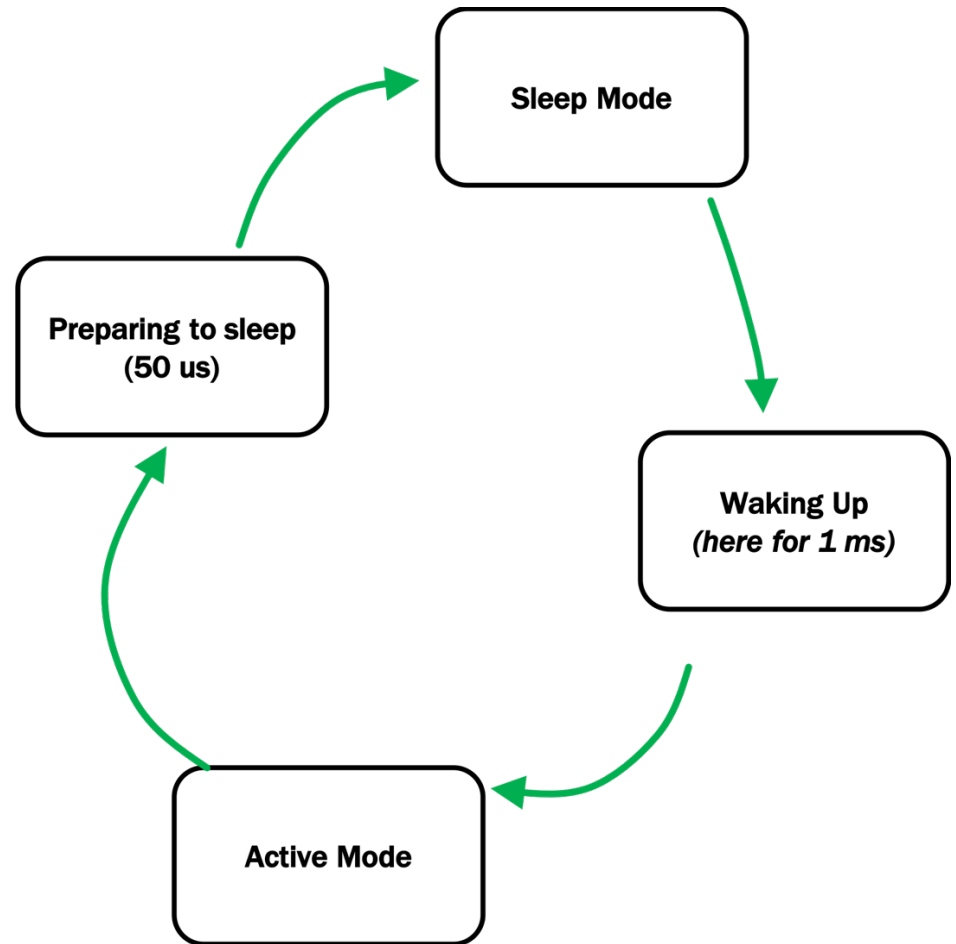
	A	B	C
Power budget			
		Read sensors	Math
MCU subsystem			
ESP32C3		28.00	28.00
LED		5.00	5.00
		33.00	33.00
Sensor subsystem			
SGP41		3.00	0.03
SHTC3		0.90	0.07
LED		5.00	5.00
I2C		0.66	0.33
		9.56	5.43
Power subsystem			
MCP73871		0.03	0.03
AP7361C		0.06	0.06
LEDs		5.00	5.00
Therm		0.05	0.05
PROG1		3.70	3.70
		8.84	8.84
cycle time (ms)		51.40	47.27
10000		70	100
		3.6	4.7
		0.4%	0.5%

Your System

- Your system is going to have a much more complicated flow chart
- Likely inner loops...outer loops.
- So you'll be duty-cycling the duty-cycles (and that's fine)

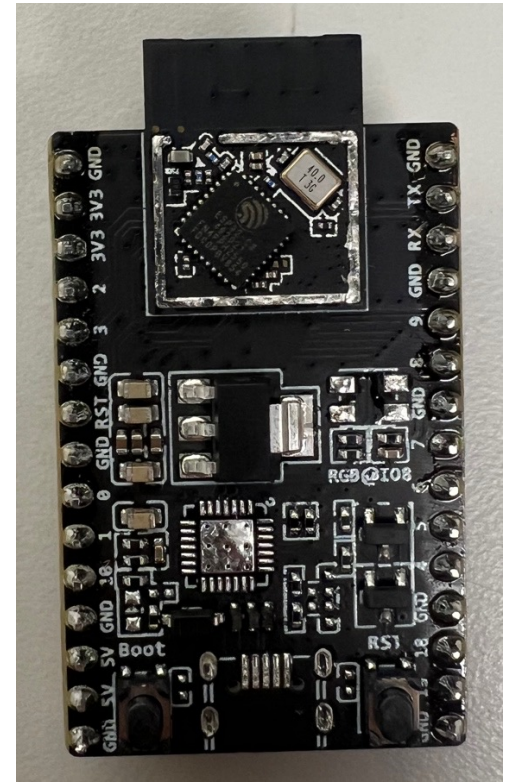
Also Keep in Mind Start-Up Costs

- If you send a device to sleep, it cannot just immediately go to sleep and then wake up and start doing stuff again. There will be shutdown and startup costs!
- In modern processors, this is usually 10,000's of clock cycles (meaning potentially milliseconds lost). So if you're going to sleep you better make sure it is worth it



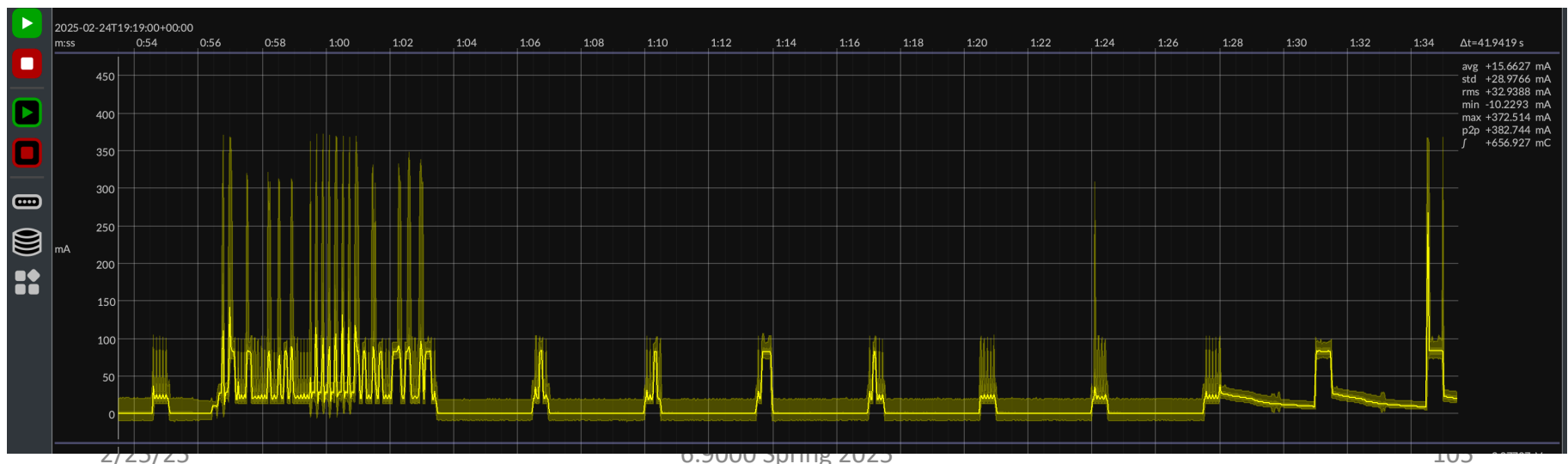
Start-Up Costs

- The ESP32 C3 can come back pretty quick (microseconds from light sleep...100's of microseconds from deep I believe)
- Certain functionalities on the ESP may be much more problematic:
 - Turning on WiFi/getting connections is costly in terms of time (and power)
 - Sending a given packet is relatively cheap
 - May motivate building up data if transferring it over WiFi, though that's limited via other things.



Plot of Sleeping, Waking, Connecting to WiFi

- Connecting to Wifi Fresh takes a lot of energy as opposed to maintaining a periodic connection
- Some nonlinear optimizations need to be figured out



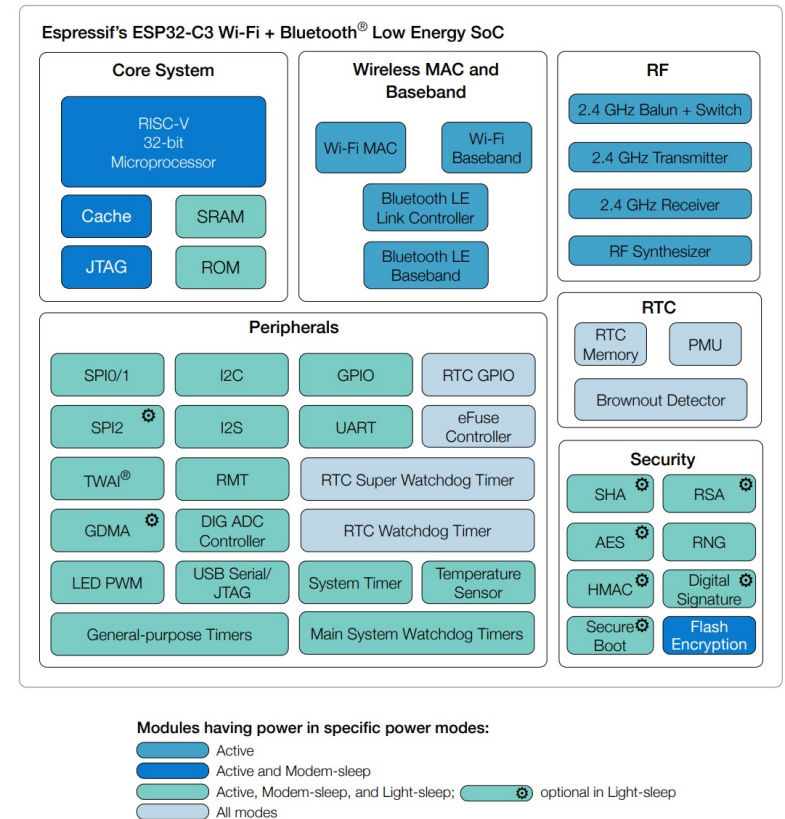
Startup Costs: Other Components



- It takes quite a few minutes for this thing to wake up and find a network successfully...that will need to be figured into everything
- Less well-documented.
- Will need to be studied and measured
- Same might go for other modules!!!

Memory

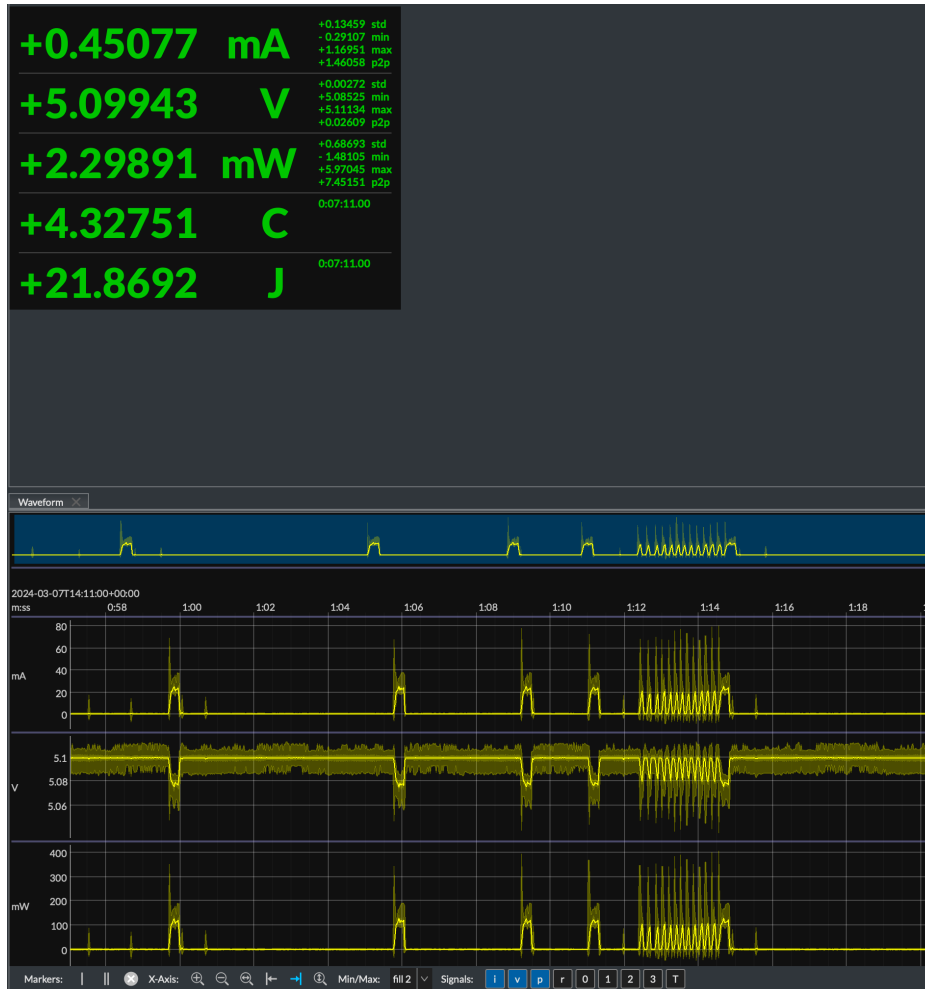
- Doing periodic sleeps can save energy, but if you're planning to gradually build up data from measurements and then periodically post it to server you have to think about how to save data during those sleeps!!!
- Light Sleep: All memory stays active
- Deep sleep only 8 kB of RTC memory is available/everything else gets reset



Making Code More Efficient

- The faster your code runs, the better it is:
 - Can turn off quicker, or can go to the next task quicker...
 - Either way bleeding less power just by being on
- This applies to on the server as well.
 - Just because the power to run your python comes from a power plant in Canada or New Jersey doesn't mean we should ignore it
 - Data centers consume an ever-increasing portion of the US energy consumption in the US (~2.0% currently)

No matter what you will then need to get actual data



- Compare your estimated power budget records to actual corrected data.
- Compare and contrast and find where the issues are!