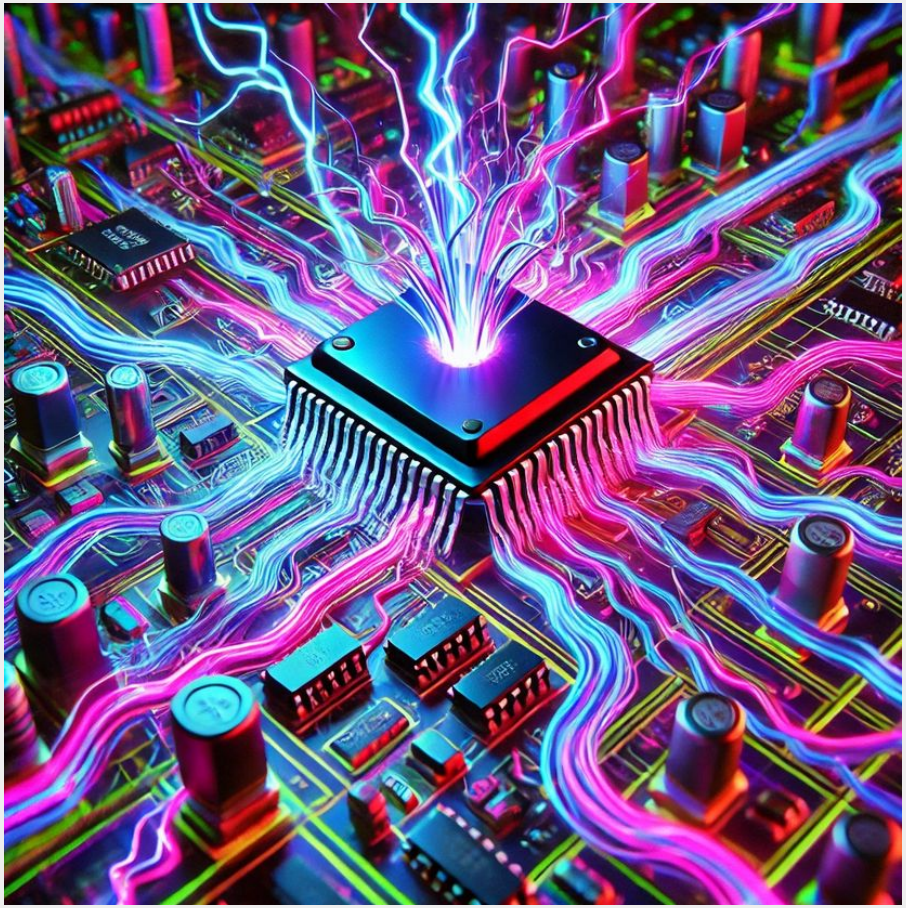
Hardware Security
Week 7 -
PCB attacks and Hardware Trojans

26T1

Hammond Pearce

Slide material acknowledgements to just me, this one is all me



RECAP

1. Why are power analysis attacks so hard to prevent?

2. What are the differences between SPA, DPA, and CPA?

3. What are some techniques to minimise the signals needed for power attacks?

4. What's the general idea behind a fault attack?

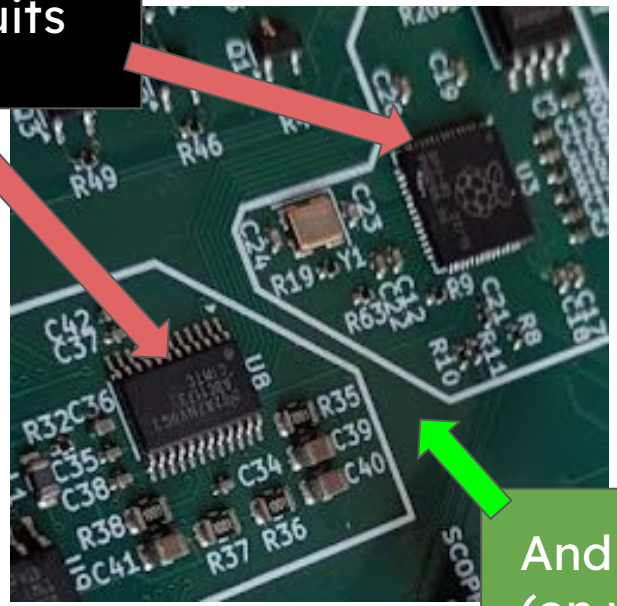
5. How can we prevent against fault attacks?

Let’s talk about Hardware

Hardware is more than just integrated circuits

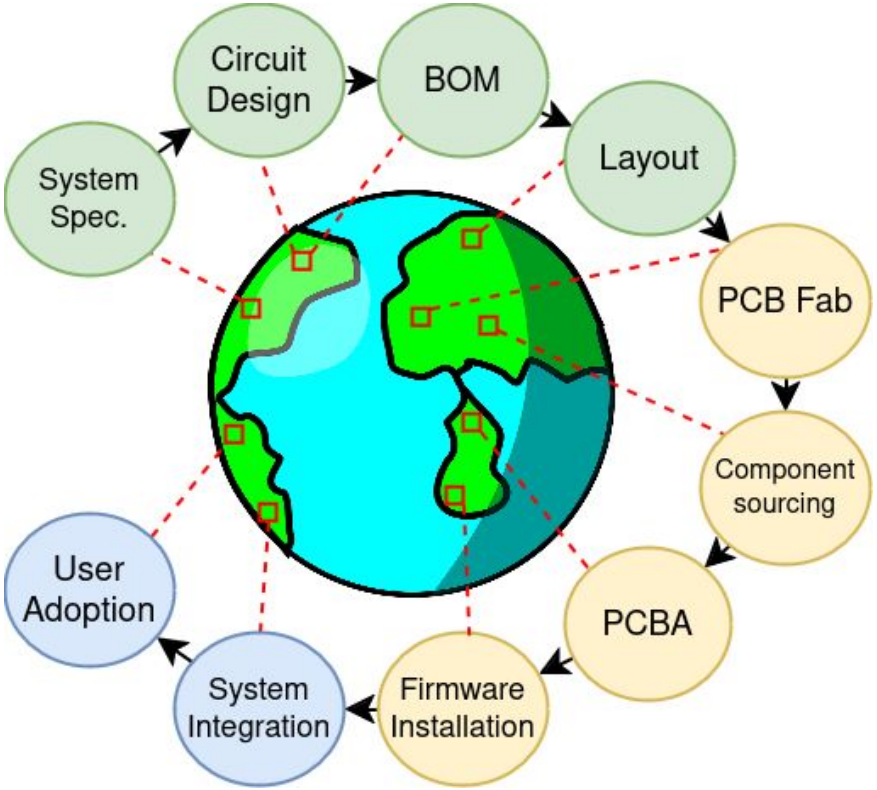
In this course, hardware is:

Integrated circuits

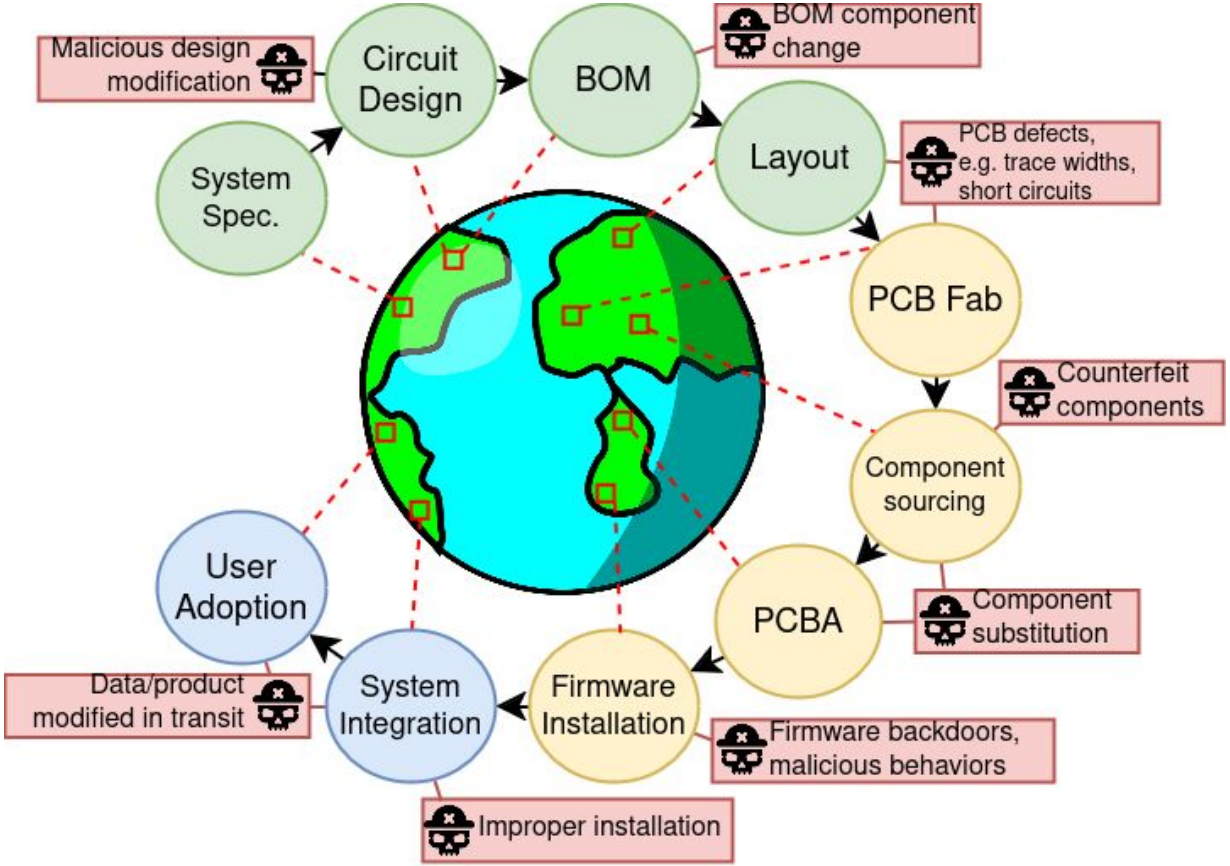


And Printed Circuit Boards
(on which they are placed)

Printed Circuit Boards also have long supply chains



Many attacks might be possible!!



Supply Chain Hardware Trojan Attacks

The Long Hack: How China Exploited a U.S. Tech Supplier

For years, U.S. investigators found tampering in products made by Super Micro Computer Inc. The company says it was never told. Neither was the public.

By Jordan Robertson and Michael Riley
February 12, 2021, 5:00 AM

FEDERAL CONTRACT OPPORTUNITY

Link Pursuit | Add Pursuit | Search

Microsystems Exploration: Safeguards against Hidden Effects and Anomalous Trojans in Hardware (SHEATH)

Pre-Solicitation | 2 Years Past Due | Inactive

Updated Jul 18 2019, 11:17 am EDT

Modchips of the State: 35th Chaos Communication Congress, 2018.
https://fahrplan.events.ccc.de/congress/2018/Fahrplan/events/9597.html

Printed Circuit Board Attack Model

- Malicious vendors
- Malicious factory
- Malicious shippers

What can they do?

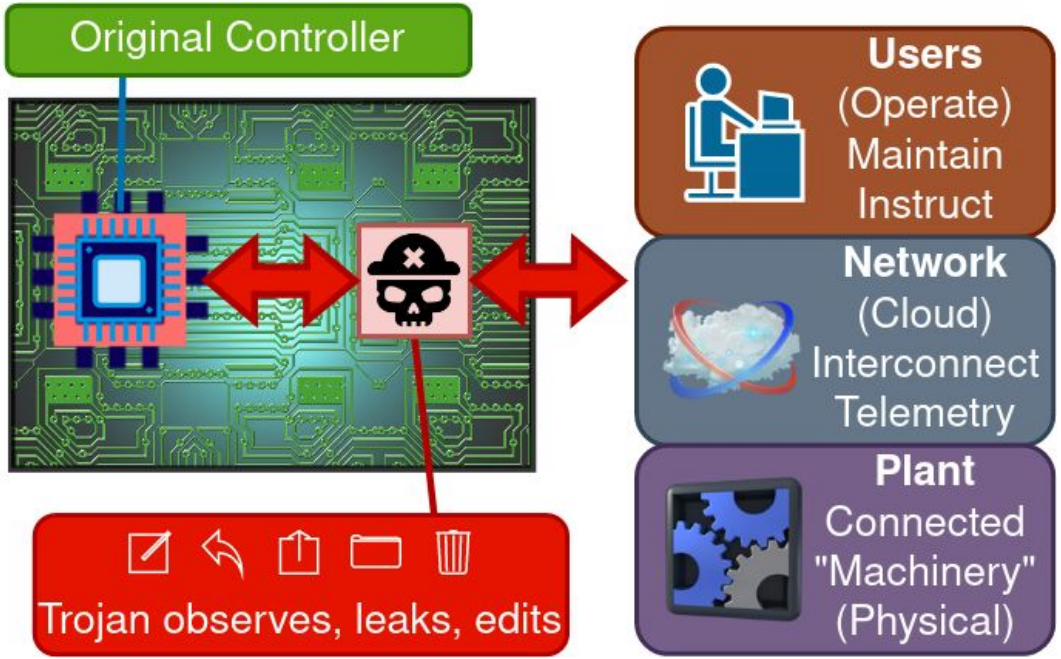
- Substitute or add components
  - Counterfeits or Trojans
  - Modchips
- Alter layouts
- Install alternative firmware

Modchip in Cisco Hardware

- Counterfeiter wants to overcome auth. checks
- Adds custom hardware to counterfeit PCBs

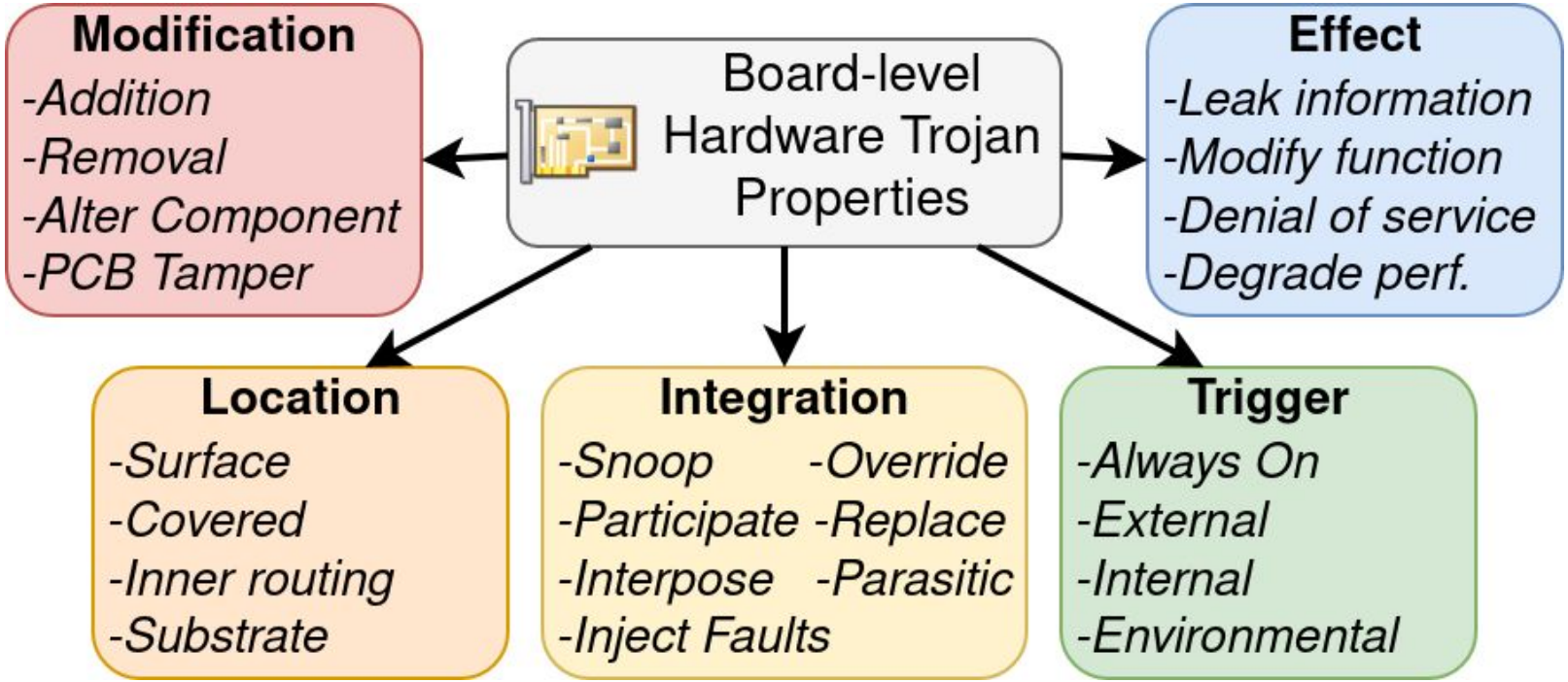
https://hackaday.com/2023/02/01/counterfeit-cisco-hardware-bypasses-security-checks-with-modchips/

Hardware Modification (Trojans): Impacts



PCB alterations can lead to complete compromise of a system!

Hardware Trojan Taxonomy



Why can these happen?

- Computer systems need to do a lot of stuff
- It’s difficult to fit all the needed functionality in a single chip
  - Also expensive and application specific
- Computer systems (built on PCBs) instead tend to:
  - Use multiple chips
  - Usually: flexible, re-usable, and well-defined component ICs
  - Peripherals!

Examples of peripheral ICs

CAT9554HV6I

8 BIT I2C AND SMBUS I/O PORT WI

QUICK VIEW

$0.37000

MAX31629MTA+T

IC I2C THERMOMETER RTC TDFN

QUICK VIEW

$3.16500

LIS2DE12TR

ACCEL 2-16G I2C/SPI 12LGA

QUICK VIEW

$1.87000

CS4340-BS

STEREO DAC FOR DIGITAL AUDIO

QUICK VIEW

$2.75000

LC75834WHS-TLM-E

IC LCD DISPLAY DRIVER

QUICK VIEW

$1.26000

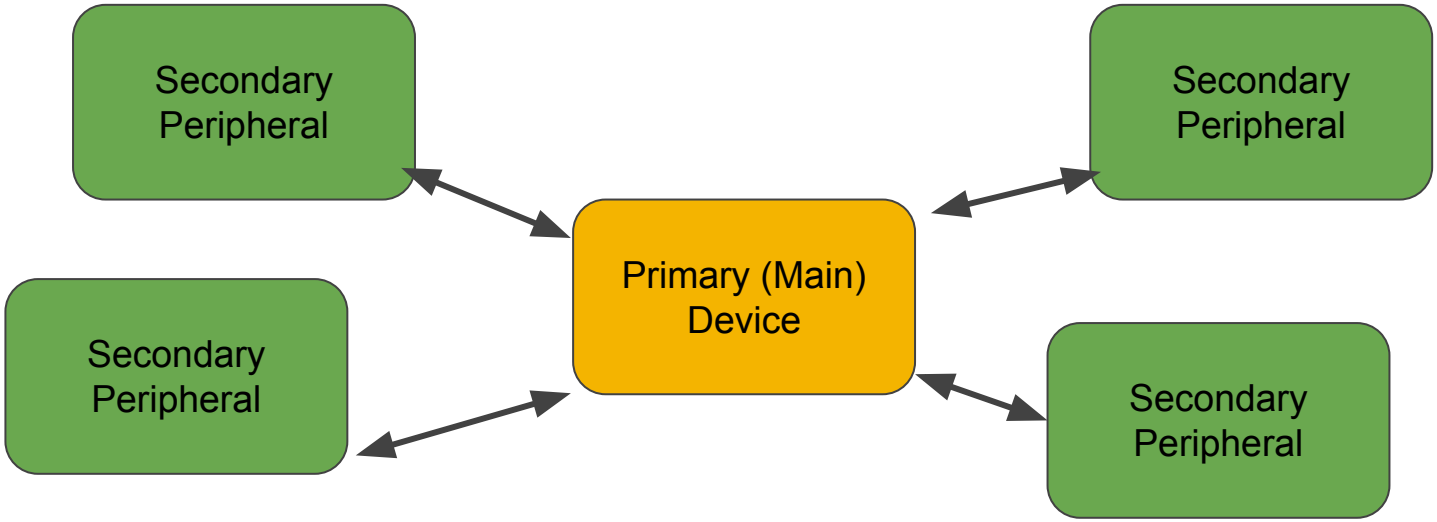
AT34C02BY5-10YU-1.7

EEPROM, 256X8, SERIAL, CMOS

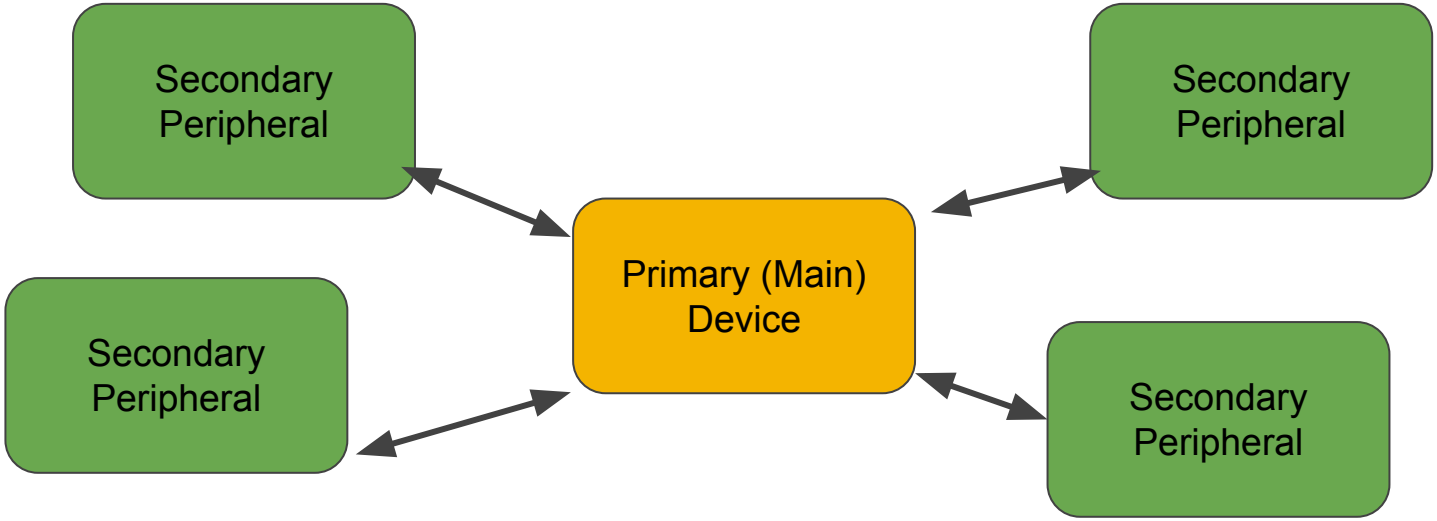
QUICK VIEW

$1.65000

Peripheral communication



Peripheral communication



Things to consider!

Needed “speed” of communication?
Needed “frequency” of communication?
“Number of communicators”?
Needed “message size”?
Needed “hardware”? (cost)
“Communication initiators”?
Other things - voltage requirements, noise resistance …

Types of communication:

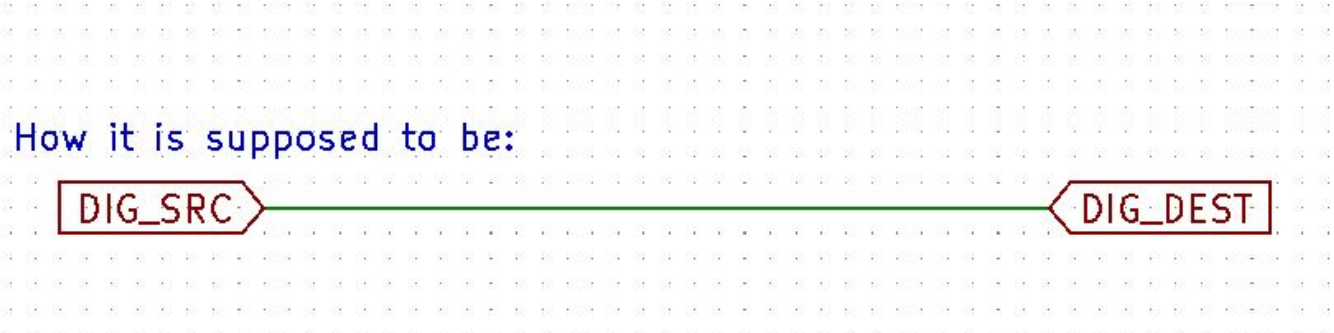
??

Types of communication:

- GPIO - individual digital wires
  - E.g. light switch
- Individual analog voltage-carrying (or current-carrying) wires
  - E.g. dimmer switch
- Serial communication (values changing over time) with some *PROTOCOL*
  - E.g……
  - USB, UART (RS-232, RS-422, RS-485), CAN
  - I2C, SMBus, SPI
  - Ethernet
- Note that we focus on the hardware-level signal protocol
  - Still, there are some hardware-level decisions open for interpretation (e.g. voltage)

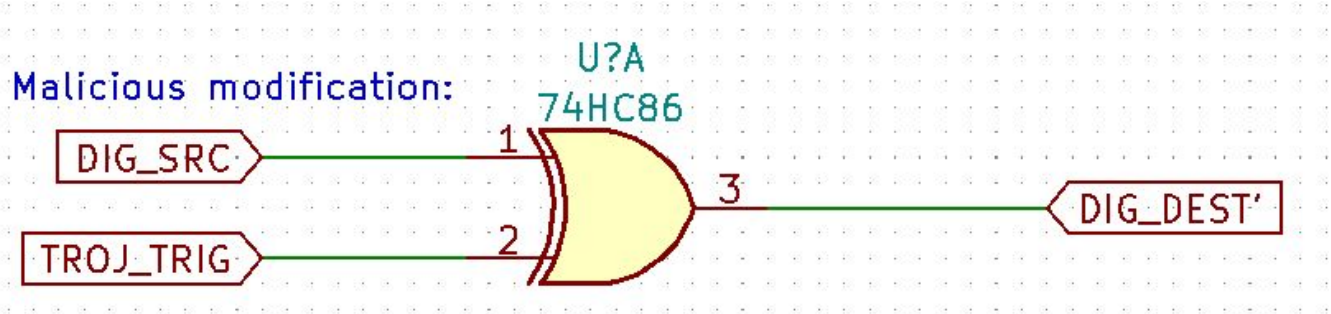
Let’s talk wires

Single Digital Wire

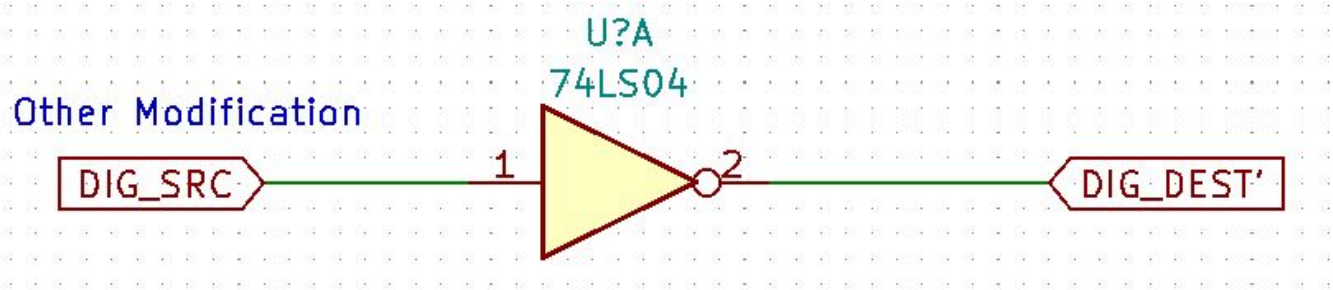


Can we interfere?

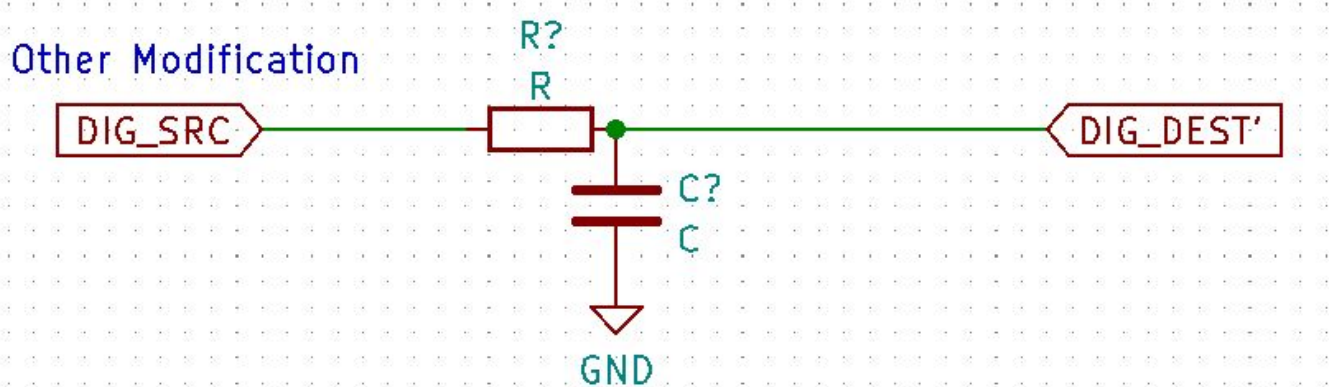
Single Digital Wire



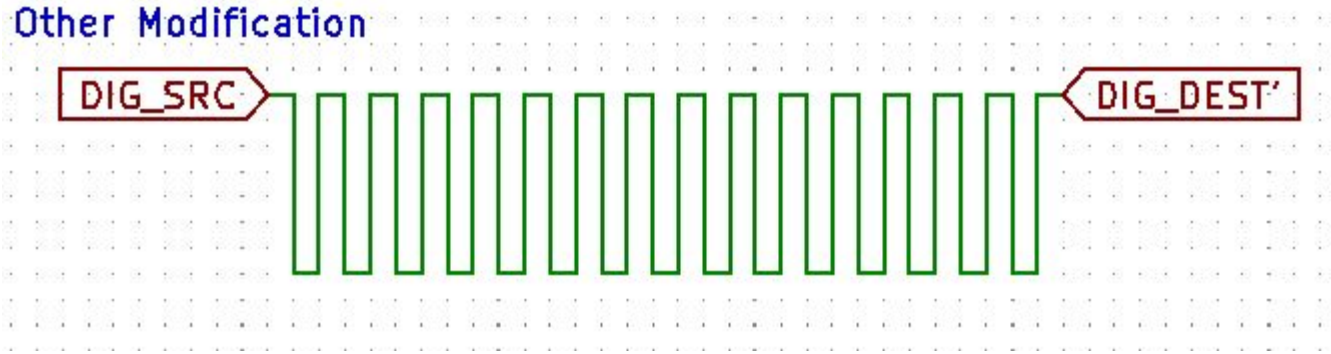
Single Digital Wire



Single Digital Wire



Single Digital Wire



Single Digital Wire

These modifications are general

- Analog wires also may have injected modifications
  - E.g. capacitor, resistors to change voltages
- Any wire of a bus protocol might also get modified
- However, targeted modifications to bus signals are more complex:
  - Undetected modification vs. bus corruption

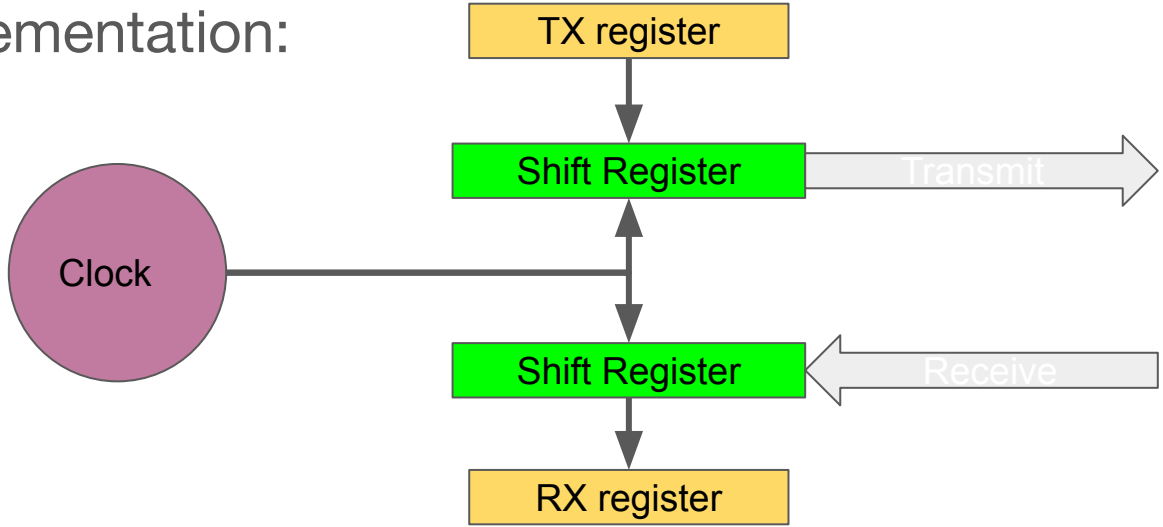
Let’s talk protocols

Basic protocol: RS-232, UART

- 1960: RS-232 (Recom. Standard 232, Electronic Industries Association)
- 1971: WD1402A, single-chip UART (prior to that, discrete implementations)
- UART is a subset of full RS-232 protocol

Basic protocol: UART (subset)

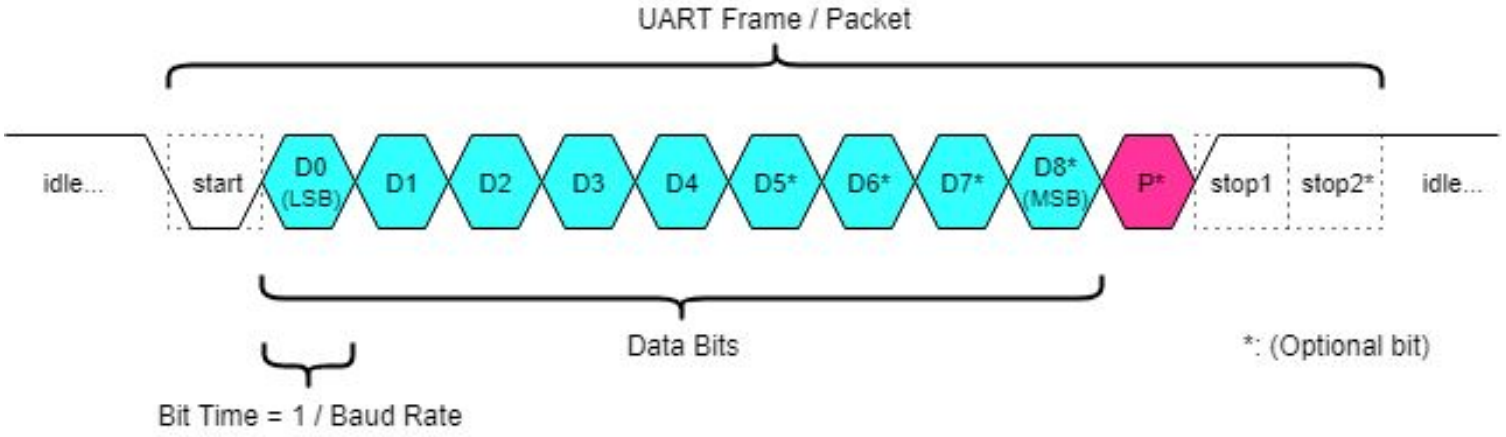
- Implementation:



- Point-to-point communication*
- Full-duplex (both devices can communicate simultaneously)

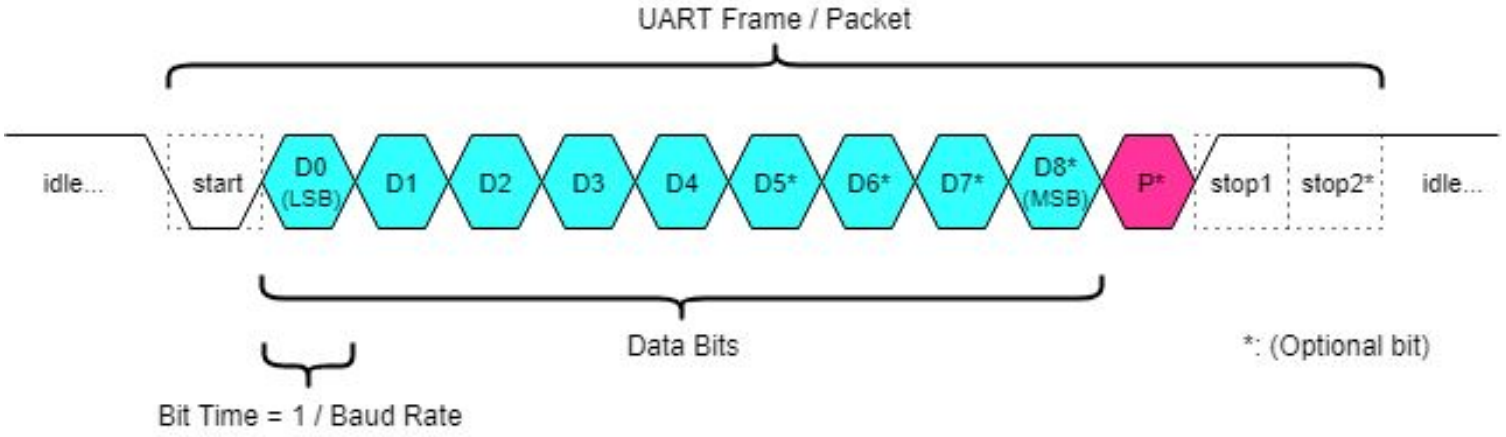
*(some revisions enable more than 2 devices...)

UART on the wire



- In-built protections (good design decisions)?

UART on the wire



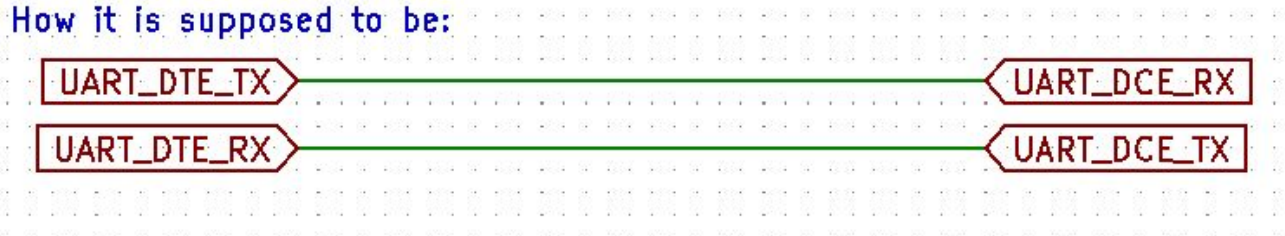
- In-built protections (good design decisions)?
  - Idle high (detect broken wire),
  - Start/Stop bits (synchronization)
  - Optional protection: extra stop bit, parity bit

UART Features / Attacking UART?

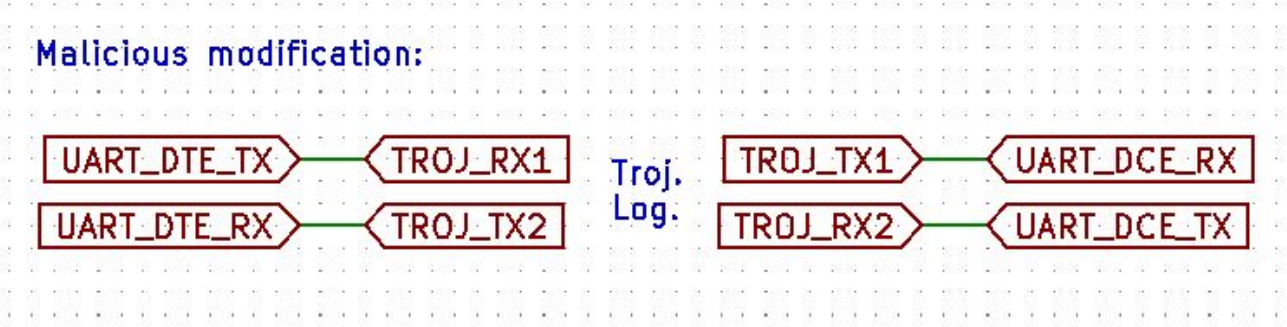
UART Features / Attacking UART?

- Weaknesses: no in-built clock (speed/sync issues)
- Parity is not encryption
- Typical application: Make request, wait, get response
  - How long to wait? Depends on application…
- Possible approach: “MitM”

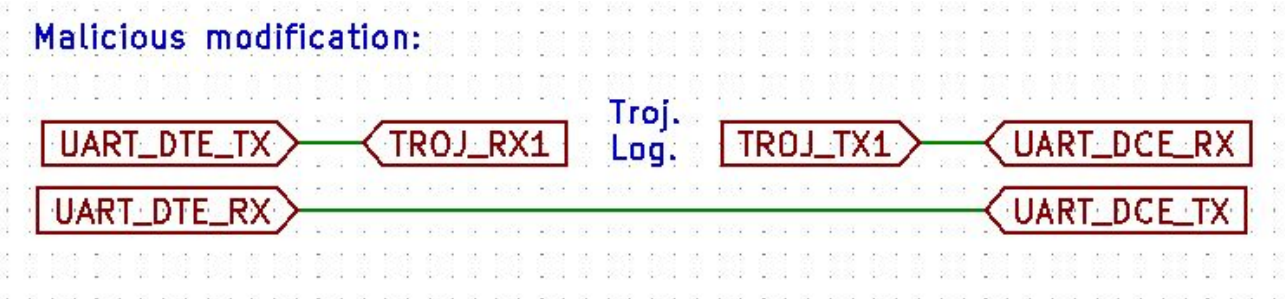
MitM UART


How it is supposed to be:
UART_DTE_TX
UART_DCE_RX
UART_DTE_RX
UART_DCE_TX

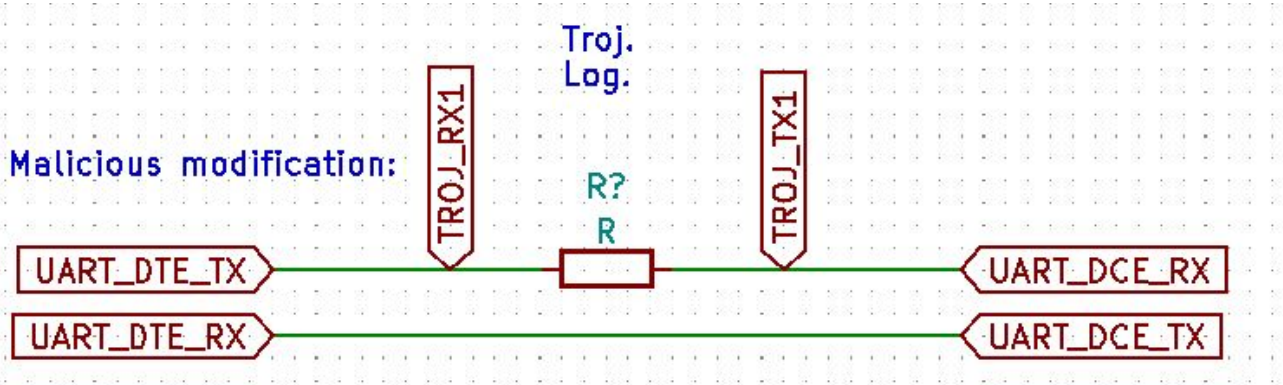
MitM UART


Malicious modification:
UART_DTE_TX
TROJ_RX1
UART_DTE_RX
TROJ_TX2
Troj.
Log.
TROJ_TX1
UART_DCE_RX
TROJ_RX2
UART_DCE_TX

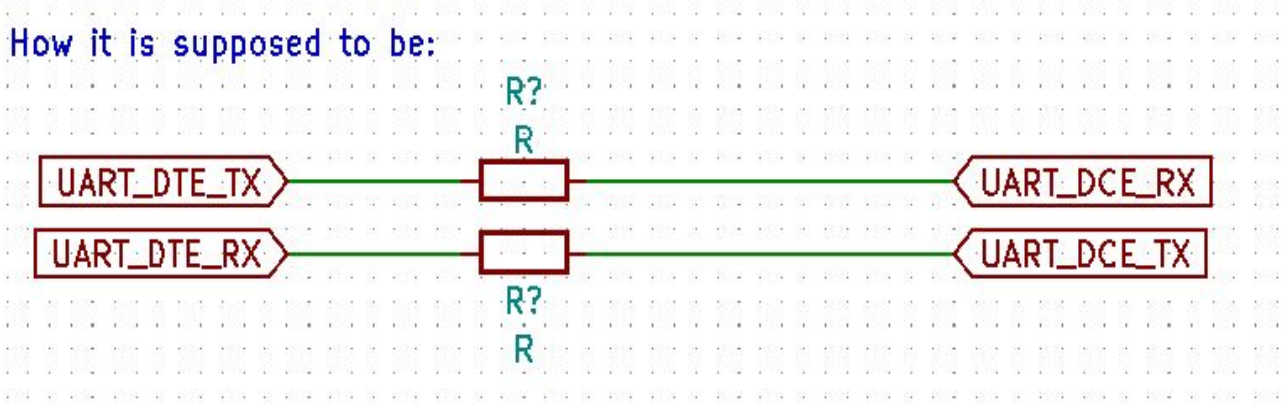
MitM UART


Malicious modification:
UART_DTE_TX
TROJ_RX1
Troj.
Log.
TROJ_TX1
UART_DCE_RX
UART_DTE_RX
UART_DCE_TX

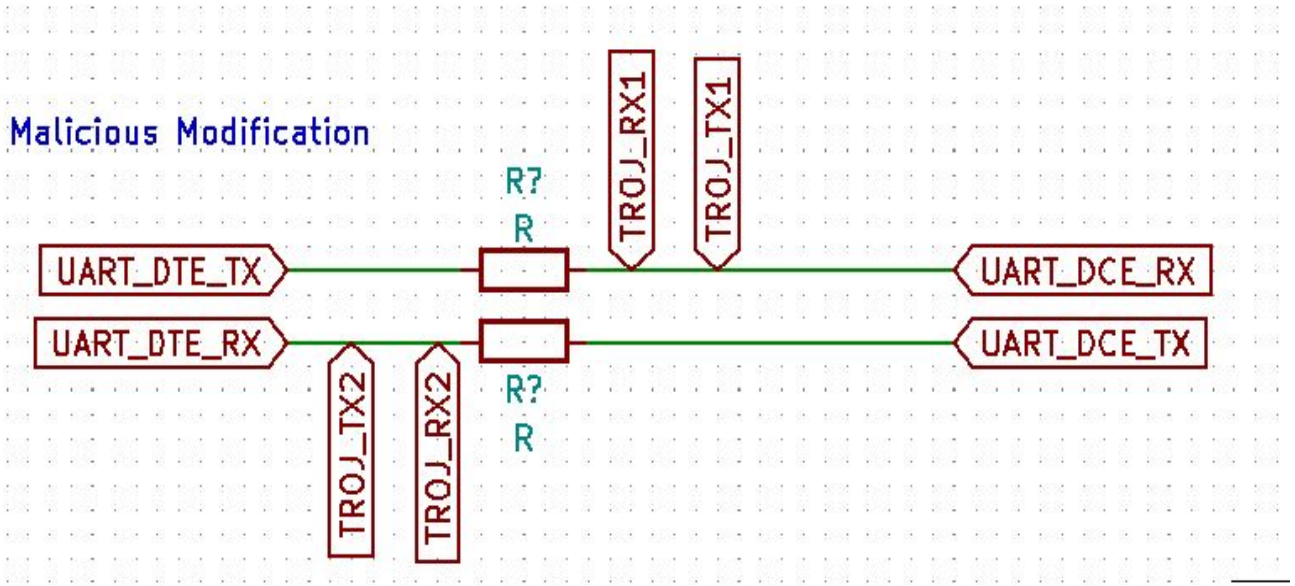
MitM UART


Troj.
Log.
Malicious modification:
TROJ_RX1
TROJ_TX1
R?
R
UART_DTE_TX
UART_DTE_RX
UART_DCE_RX
UART_DCE_TX

MitM UART


How it is supposed to be:
R?
R
UART_DTE_TX
UART_DTE_RX
UART_DCE_RX
UART_DCE_TX
R?
R

MitM UART

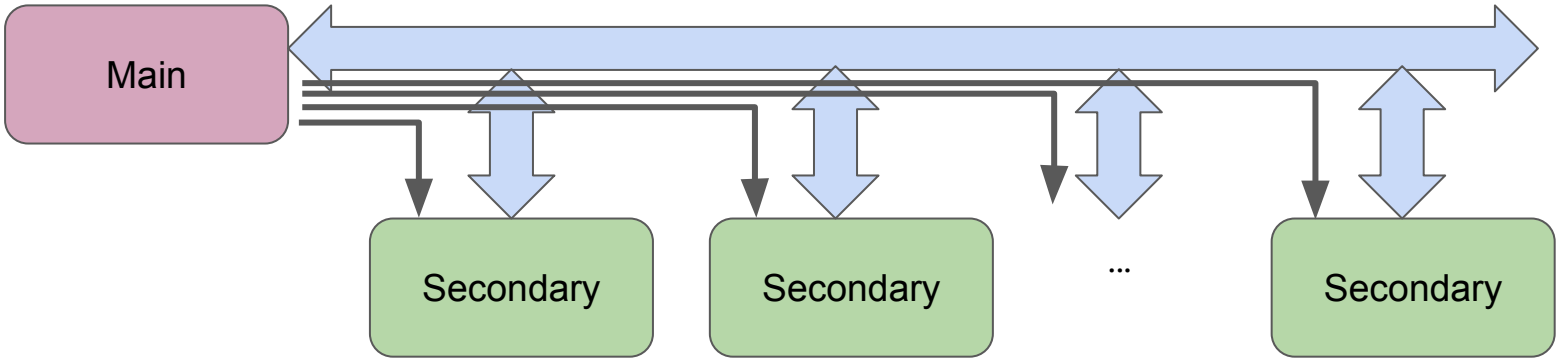

Malicious Modification
R?
R
TROJ_RX1
TROJ_TX1
UART_DTE_TX
UART_DCE_RX
UART_DTE_RX
UART_DCE_TX
TROJ_TX2
TROJ_RX2
R?
R

Other UART attacks

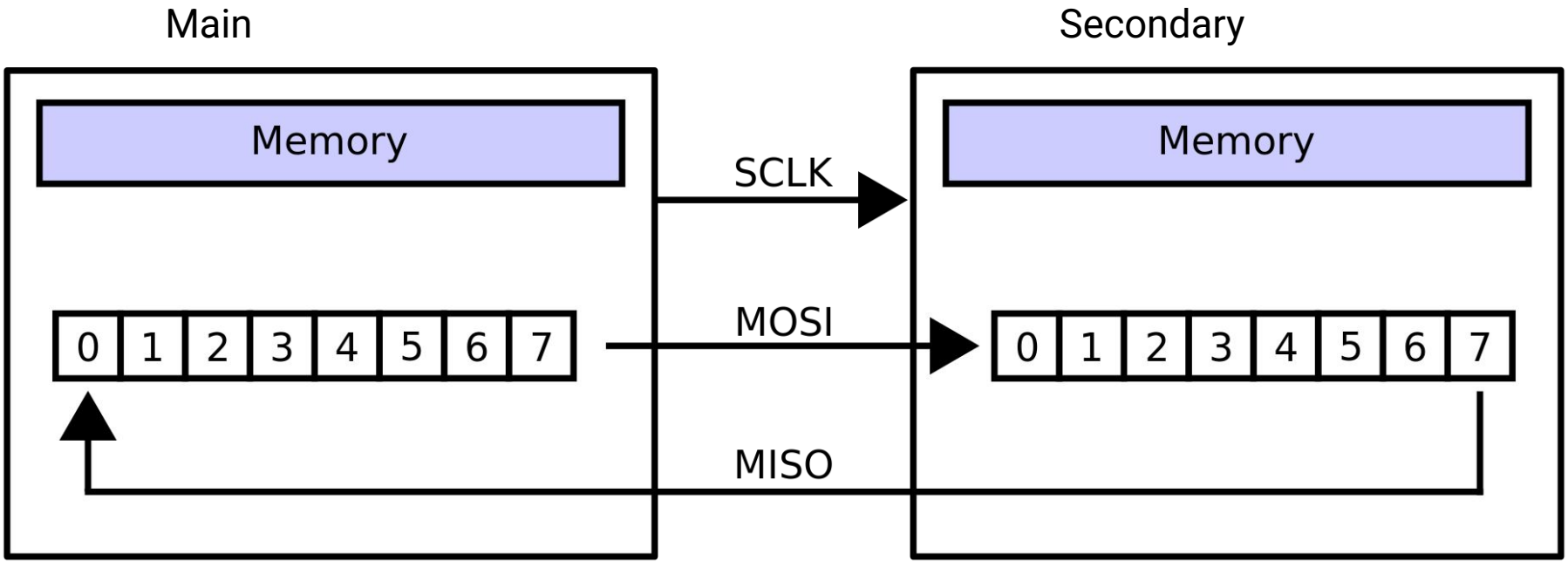
- Denial of service - “stuck at” faults
- Noise introduction / EMI leakage
- Signal re-routing
- Trace widths
- Component value changes (capacitors for resistors, different values…)
    - Speed implications

SPI

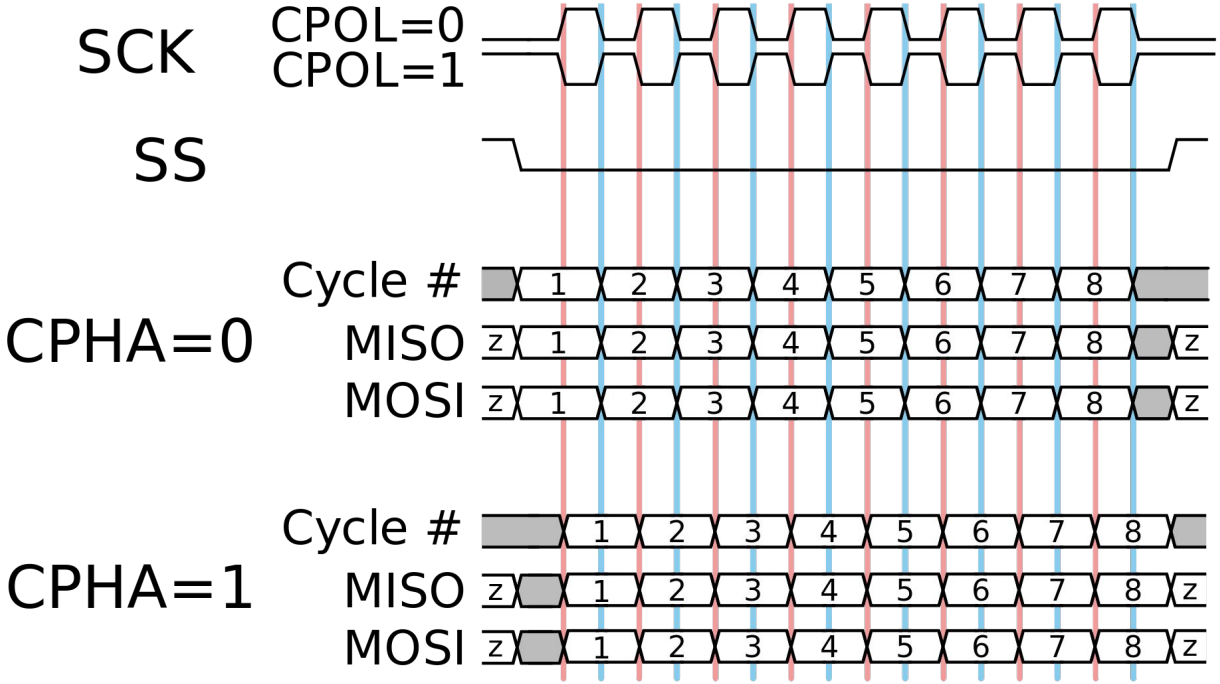
- Developed 1979 (Motorolla)
- Often used for high-bandwidth peripherals (e.g. memory, cameras)
- Four wire + some simple hardware, single “main” node, many “secondary” peripheral nodes.
- Full duplex



SPI shift registers



SPI timing



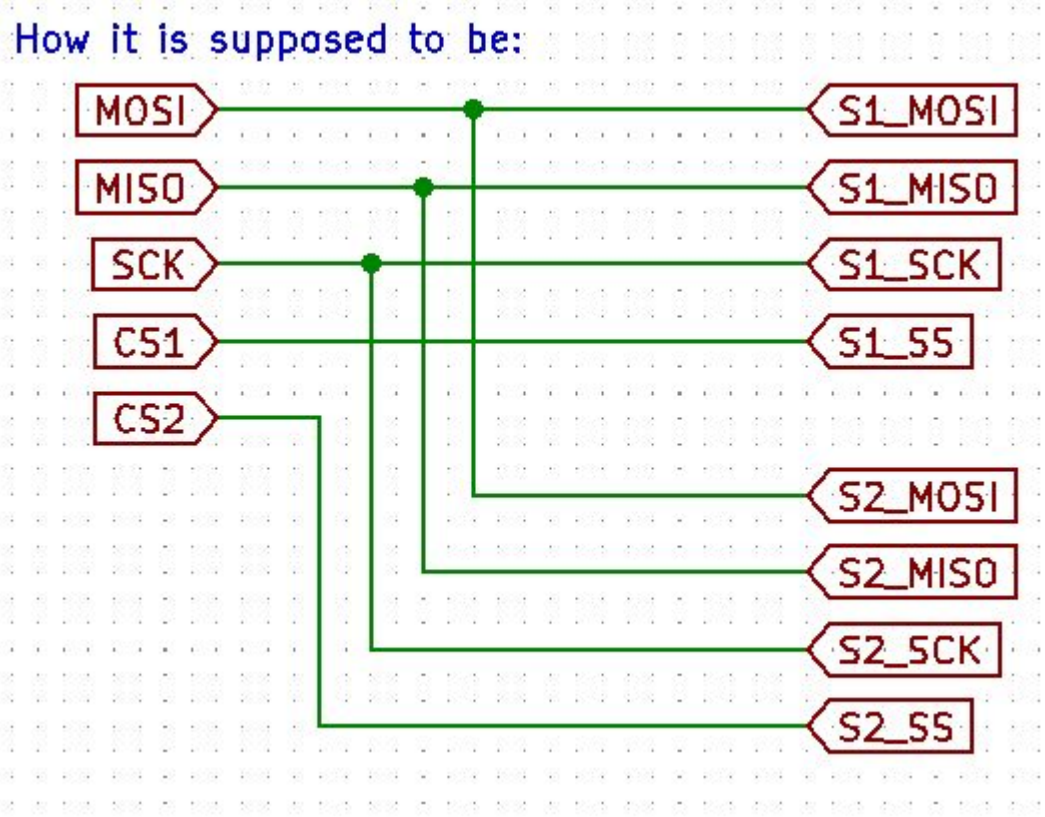
By SPI_timing_diagram.svg: en:User:Cburnettderivative work: Jordsan (talk) - SPI_timing_diagram.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=11405368

SPI Features / Attacking SPI?

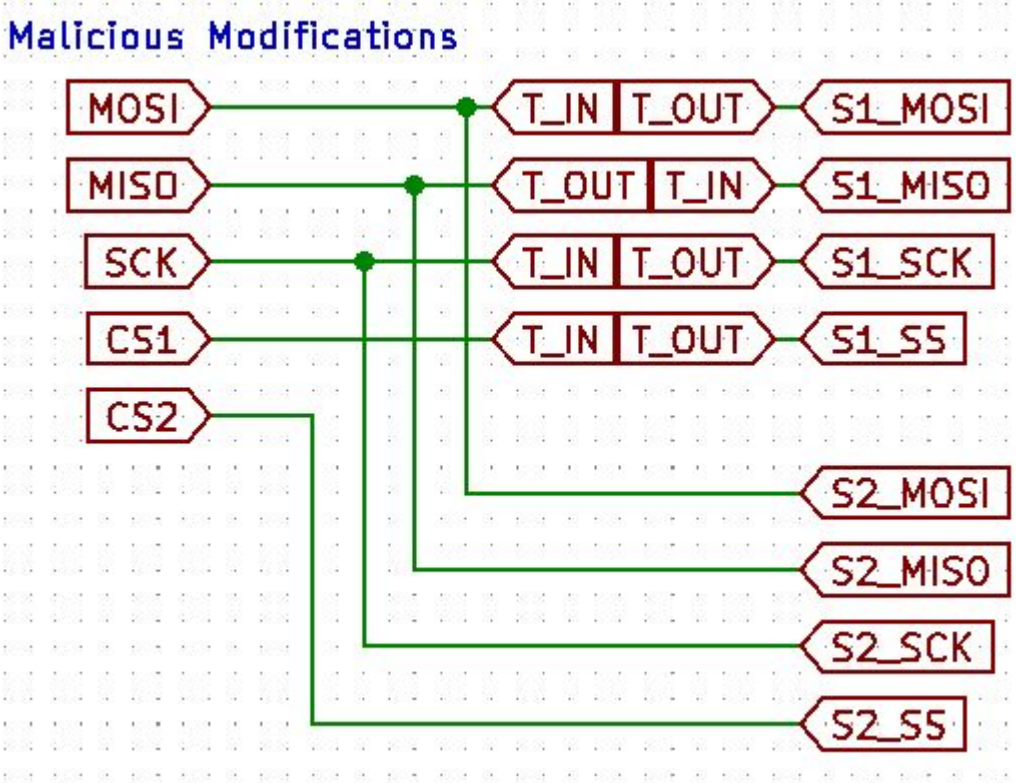
SPI Features / Attacking SPI?

- No built-in data protection of any kind (must be done in software)
- No built-in message receipt acknowledgement
- Typical lifecycle:
  - Memory: (single request) make request, start receiving back data
  - Peripheral: (multiple request) set config, wait for time/alert pin, make request, rx data
- Possible weaknesses:
  - Select pins - MitM on these pins

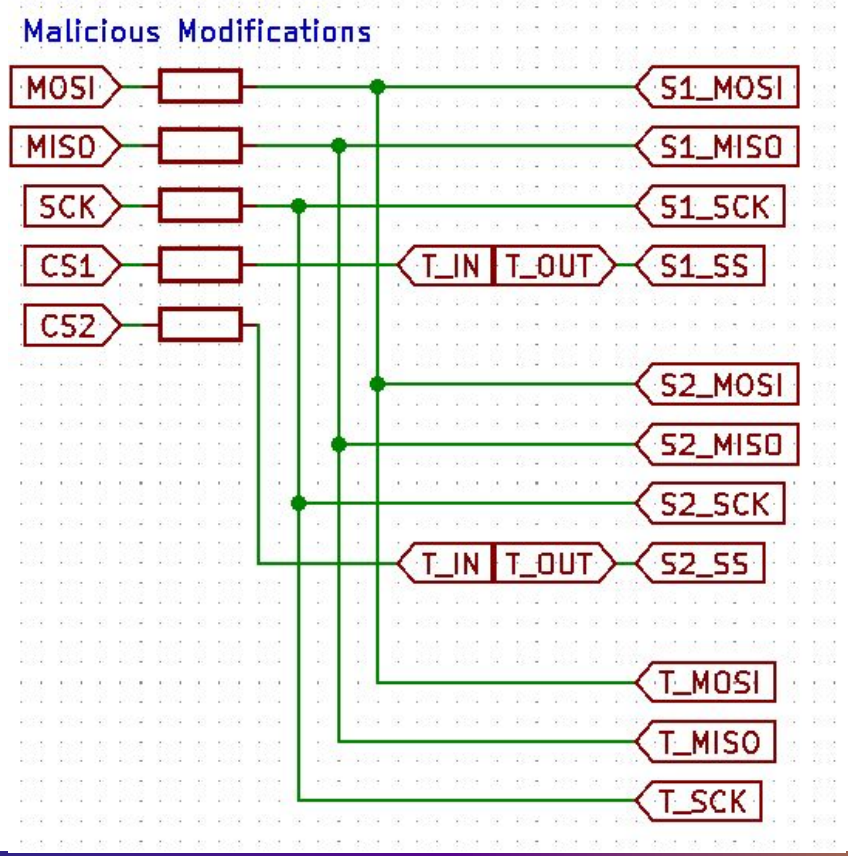
SPI weaknesses / Attacking SPI?



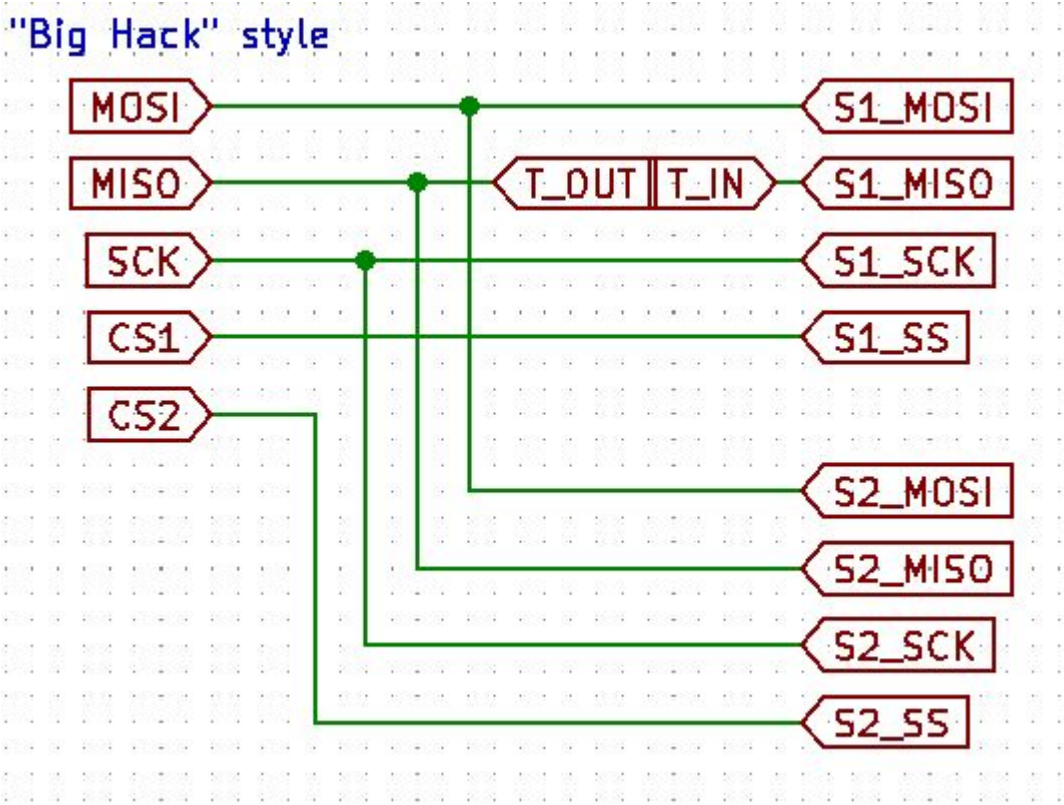
SPI weaknesses / Attacking SPI?



SPI weaknesses / Attacking SPI?



SPI weaknesses / Attacking SPI?

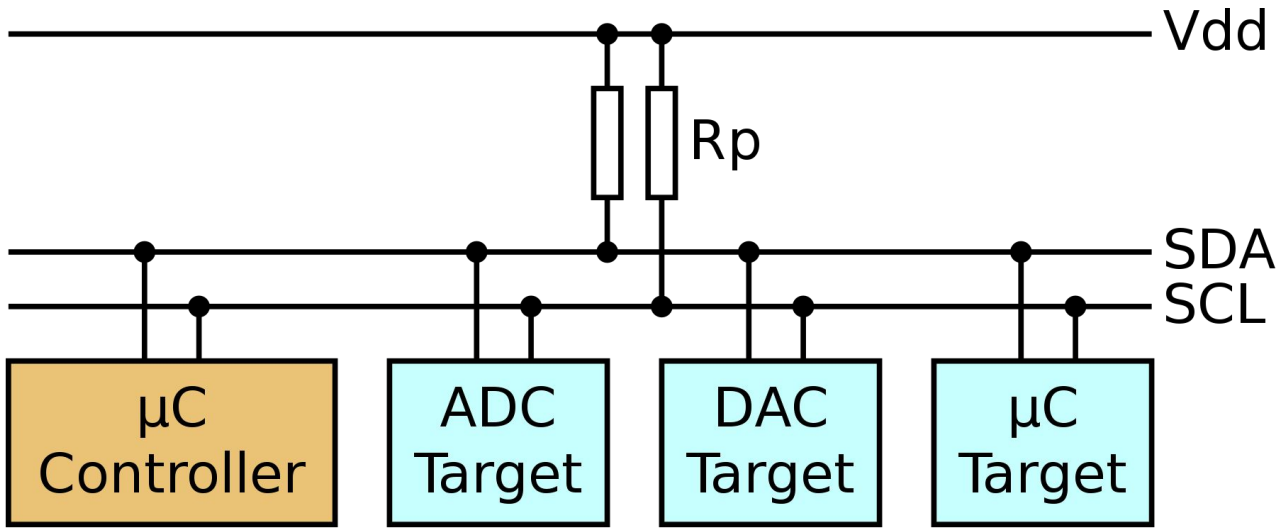


Other SPI attacks

- Similar to UART
- “Stuck at”, electrical EMI ….
- Request pins (not in our diagrams) - block?

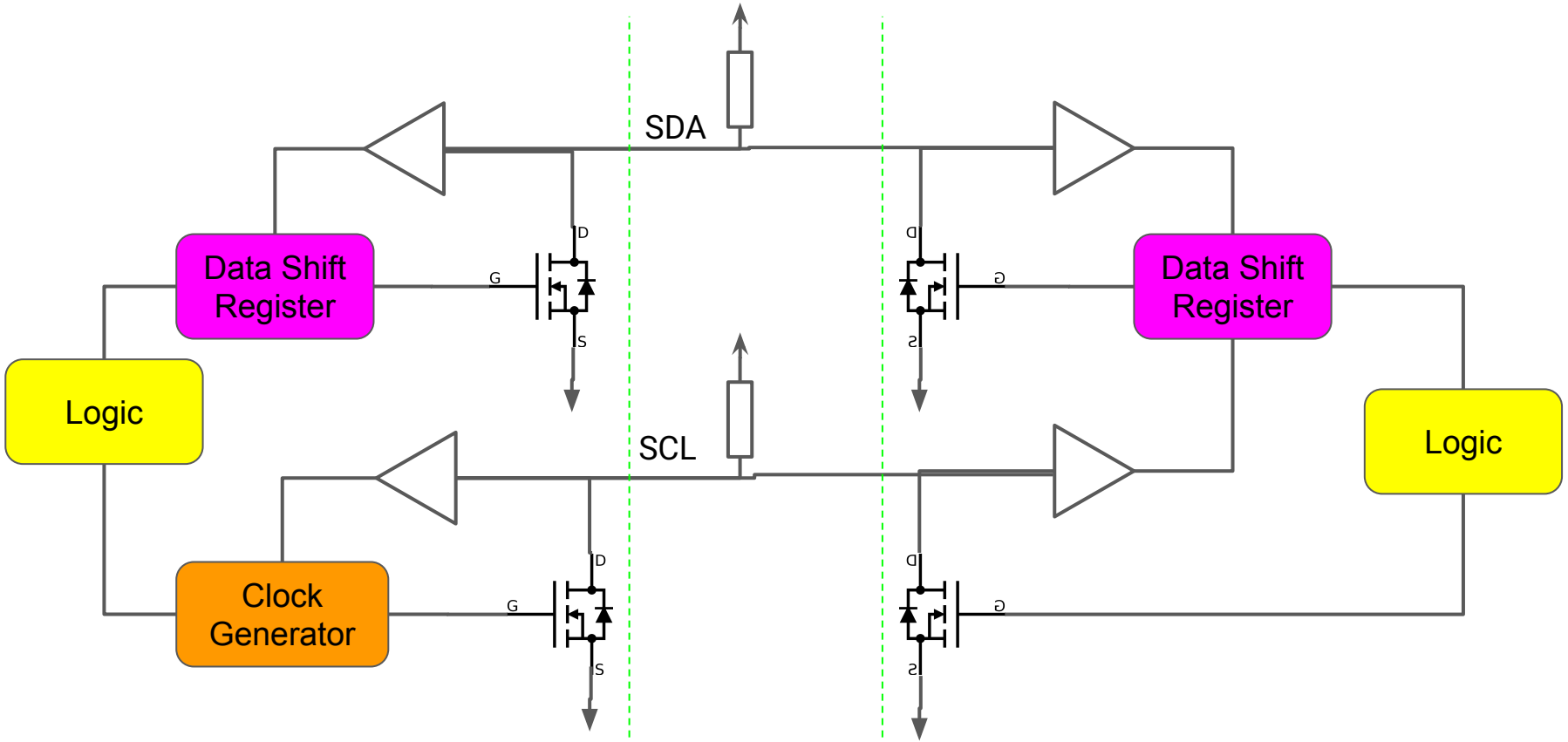
I2C / IIC

- Developed 1982 (Philips Semiconductor / “NXP”)
- Two-wire communication, half duplex



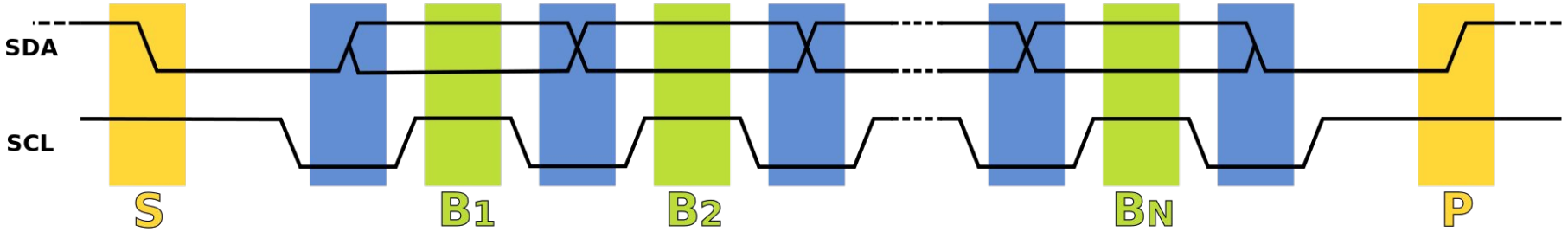
By Tim Mathias - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=111975651

I2C Shift Registers



I2C Transaction

By Marcin Floryan - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1647146



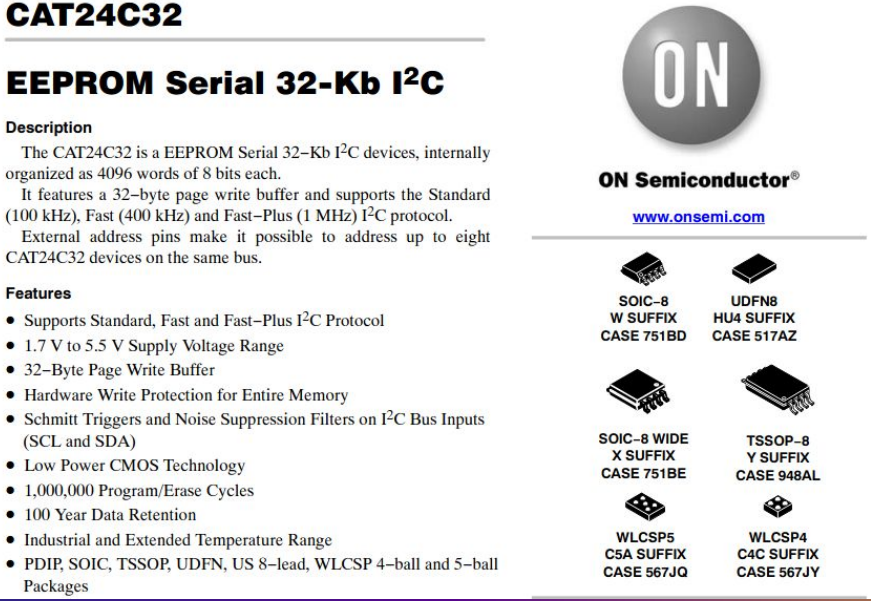
- Start bit
- Data Bits
- StoP bits

I2C Transactions

- Unlike SPI, I2C begins a transaction with the target device address
  - Usually 7 bits + R/W bit
- Every byte is acknowledged by the secondary peripheral
  - Pulls SDA low for the 9th bit
- Example transaction:
  - [“start”] [7-bit address] [“write”] [“ack”] [8-bit data] [“ack”] [“stop”]
- Clever features allow for complex behaviors, e.g. most I2C memory:
  - [“start”] [7-bit address] [“write”] [“ack”] [8-bit MEMH] [“ack”] [8-bit MEML] [“ack”]
    [“start”] [7-bit address] [“read”/”write”] [“ack”] [8-bit value at MEMH/MEML] [“ack”] [“stop”]

I2C example

Example real-world hardware (CAT24C32 EEPROM I2C):
https://www.onsemi.com/pdf/datasheet/cat24c32-d.pdf



CAT24C32

EEPROM Serial 32-Kb I²C

Description

The CAT24C32 is a EEPROM Serial 32−Kb I²C devices, internally organized as 4096 words of 8 bits each.

It features a 32−byte page write buffer and supports the Standard (100 kHz), Fast (400 kHz) and Fast−Plus (1 MHz) I²C protocol.

External address pins make it possible to address up to eight CAT24C32 devices on the same bus.

Features

- Supports Standard, Fast and Fast−Plus I²C Protocol
- 1.7 V to 5.5 V Supply Voltage Range
- 32−Byte Page Write Buffer
- Hardware Write Protection for Entire Memory
- Schmitt Triggers and Noise Suppression Filters on I²C Bus Inputs (SCL and SDA)
- Low Power CMOS Technology
- 1,000,000 Program/Erase Cycles
- 100 Year Data Retention
- Industrial and Extended Temperature Range
- PDIP, SOIC, TSSOP, UDFN, US 8−lead, WLCSP 4−ball and 5−ball Packages

ON Semiconductor®

www.onsemi.com

SOIC−8 W SUFFIX CASE 751BD

UDFN8 HU4 SUFFIX CASE 517AZ

SOIC−8 WIDE X SUFFIX CASE 751BE

TSSOP−8 Y SUFFIX CASE 948AL

WLCSP5 C5A SUFFIX CASE 567JQ

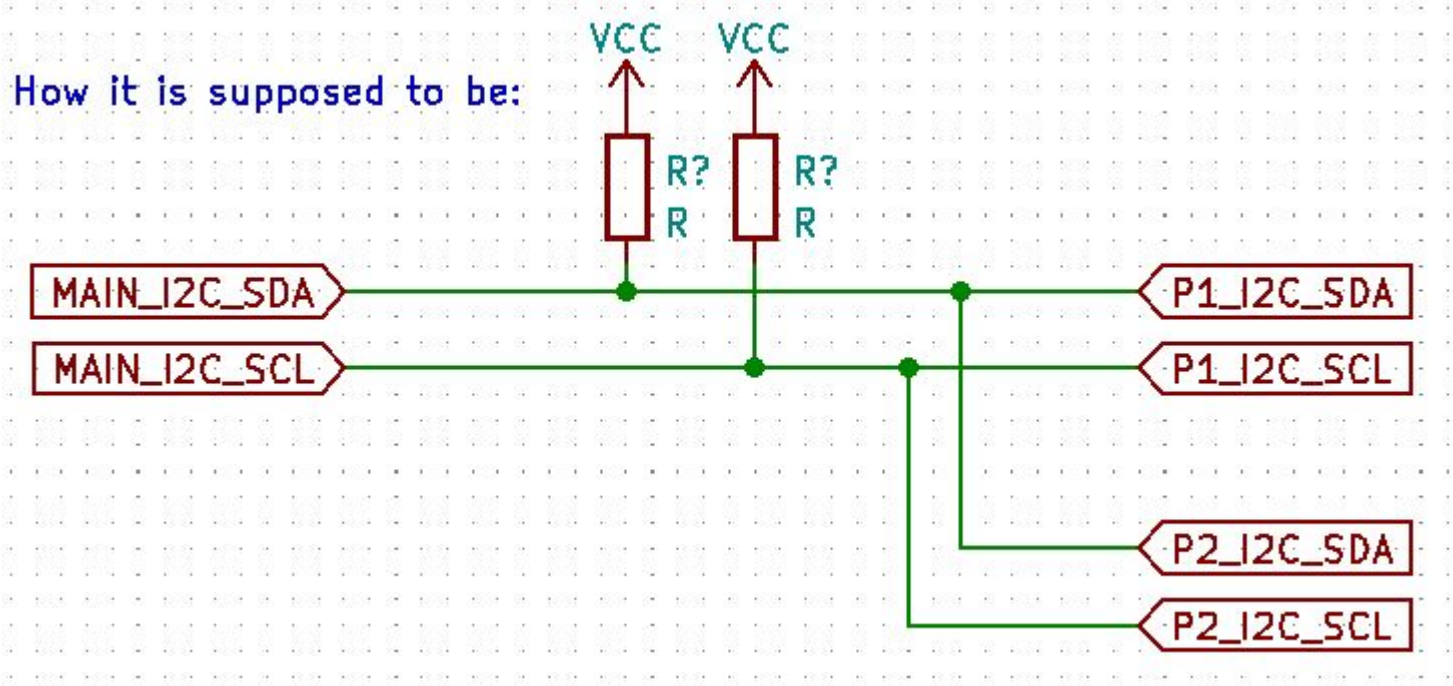
WLCSP4 C4C SUFFIX CASE 567JY

I2C Features / Attacking I2C?

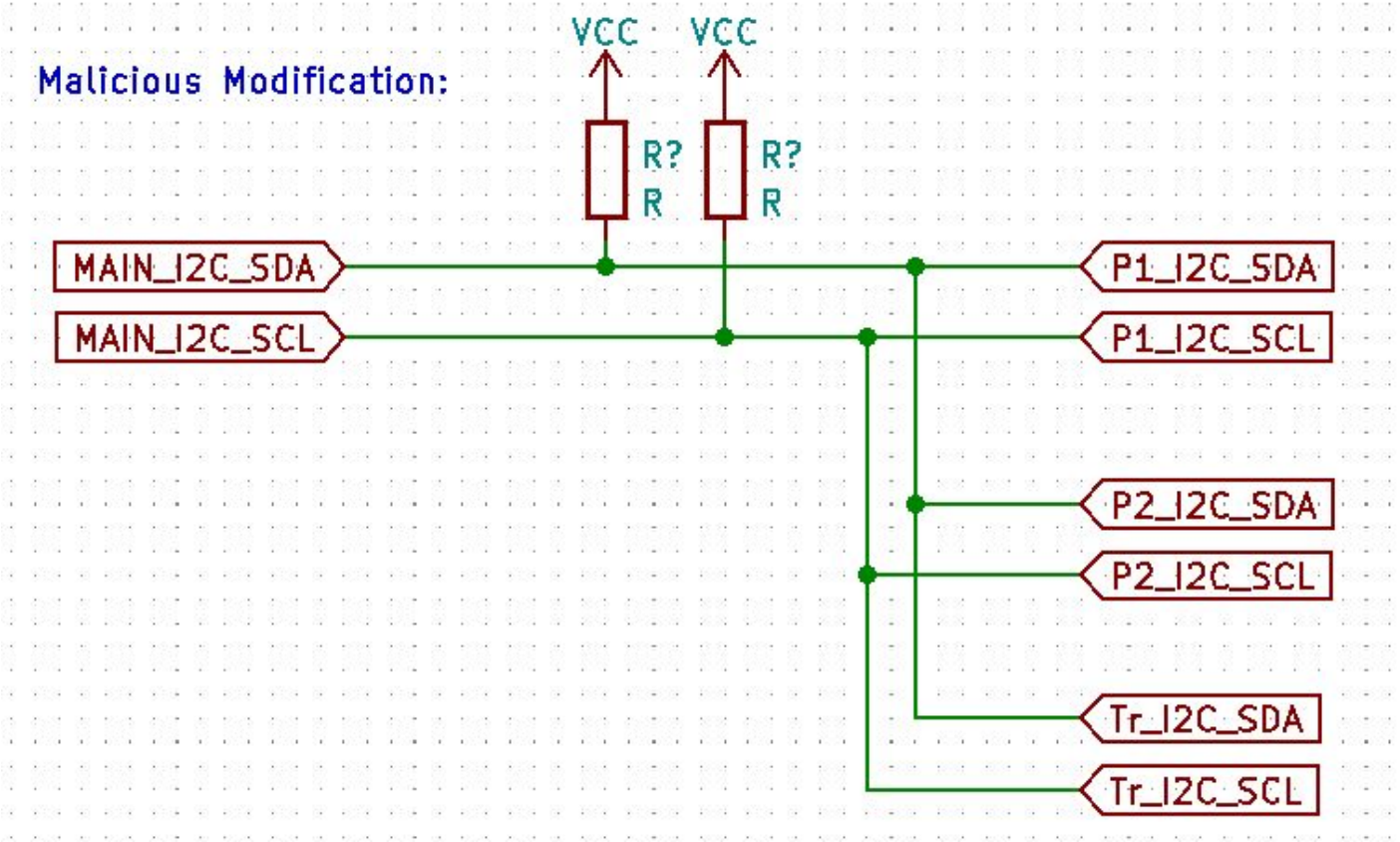
I2C Features / Attacking I2C?

- Unlike SPI/UART, I2C supports “hot-plugging” (adding extra devices)
- No protection for any device acting like Main
  - Some applications may rely on multiple Main devices!
- Some protection to detect if device goes offline
  - “Ack”/”Nack”
- However, two attacks:
  - Second “Main” device
  - “Clock stretching” enables trivial MitM!

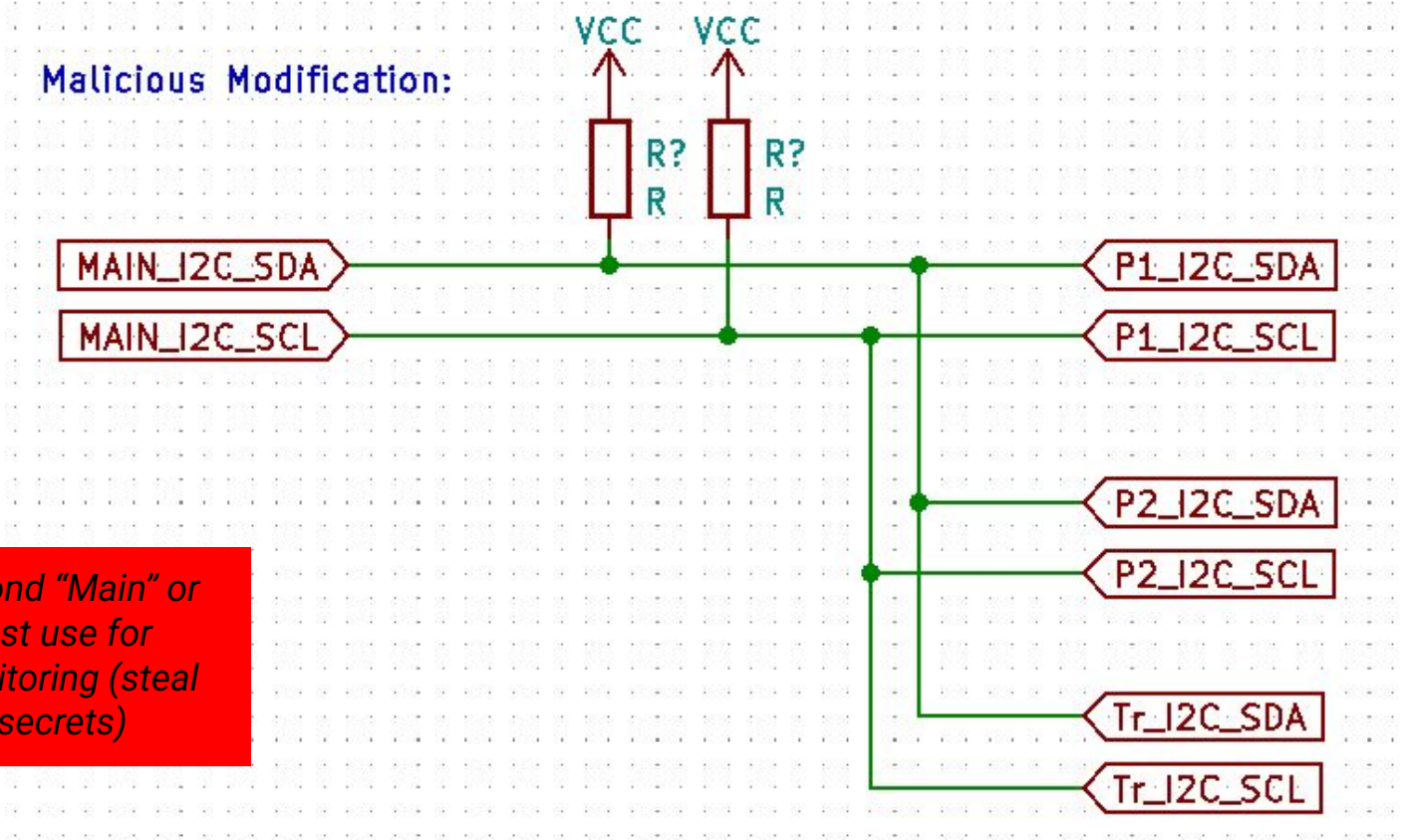
I2C Weaknesses / Attacking I2C?



I2C Weaknesses / Attacking I2C?

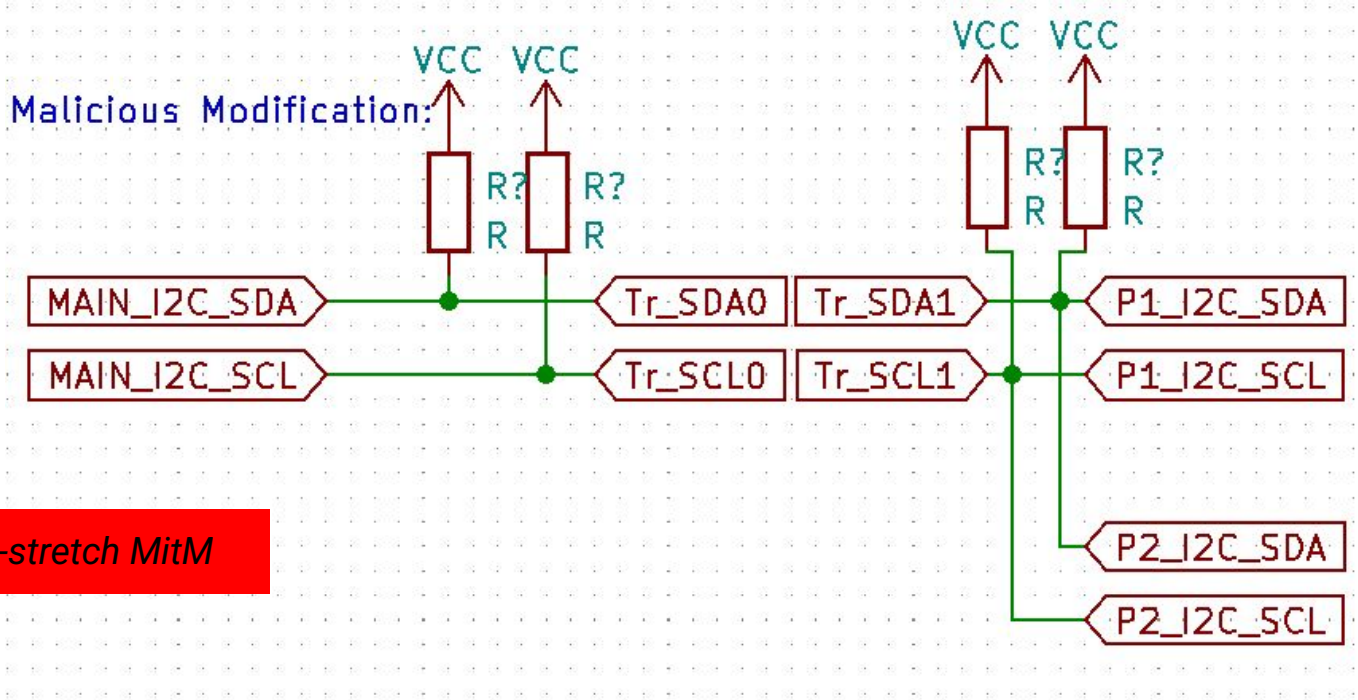


I2C Weaknesses / Attacking I2C?



Second "Main" or just use for monitoring (steal secrets)

I2C Weaknesses / Attacking I2C?

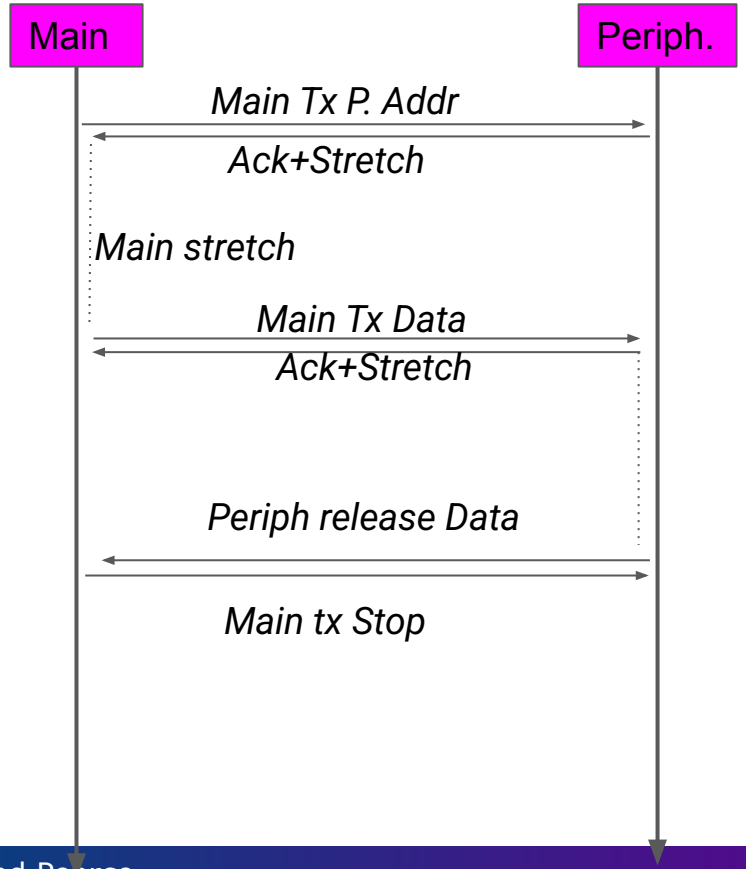


Clock-stretch MitM

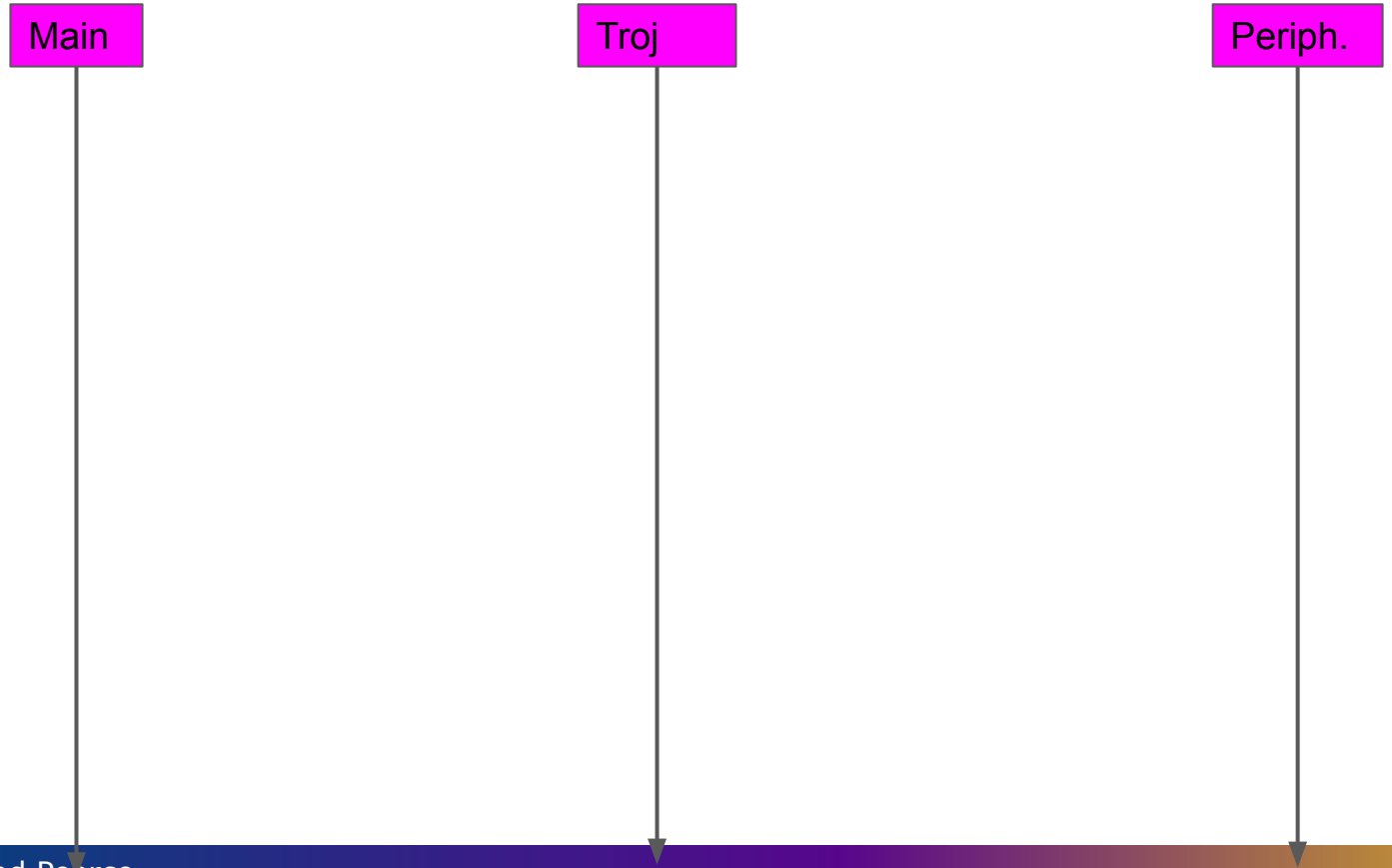
Process for a “clock-stretch” MitM attack

- Trojan located between Main device and all secondary peripheral devices
- Trojan “cuts” the I2C wires
- Trojan awaits requests from Main.
- Upon receiving request, it acknowledges but *holds Main clock low (stretch).*
- It then begins to copy the request to the secondary peripheral devices.
    - Upon receiving the acknowledgement from the peripheral, it stretches peripheral clock.
- It awaits command from Main, then stretches once finished.
    - Upon receiving command, it edits or relays it to the peripherals, and stretches.
- This continues on and on until the command from Main finishes.

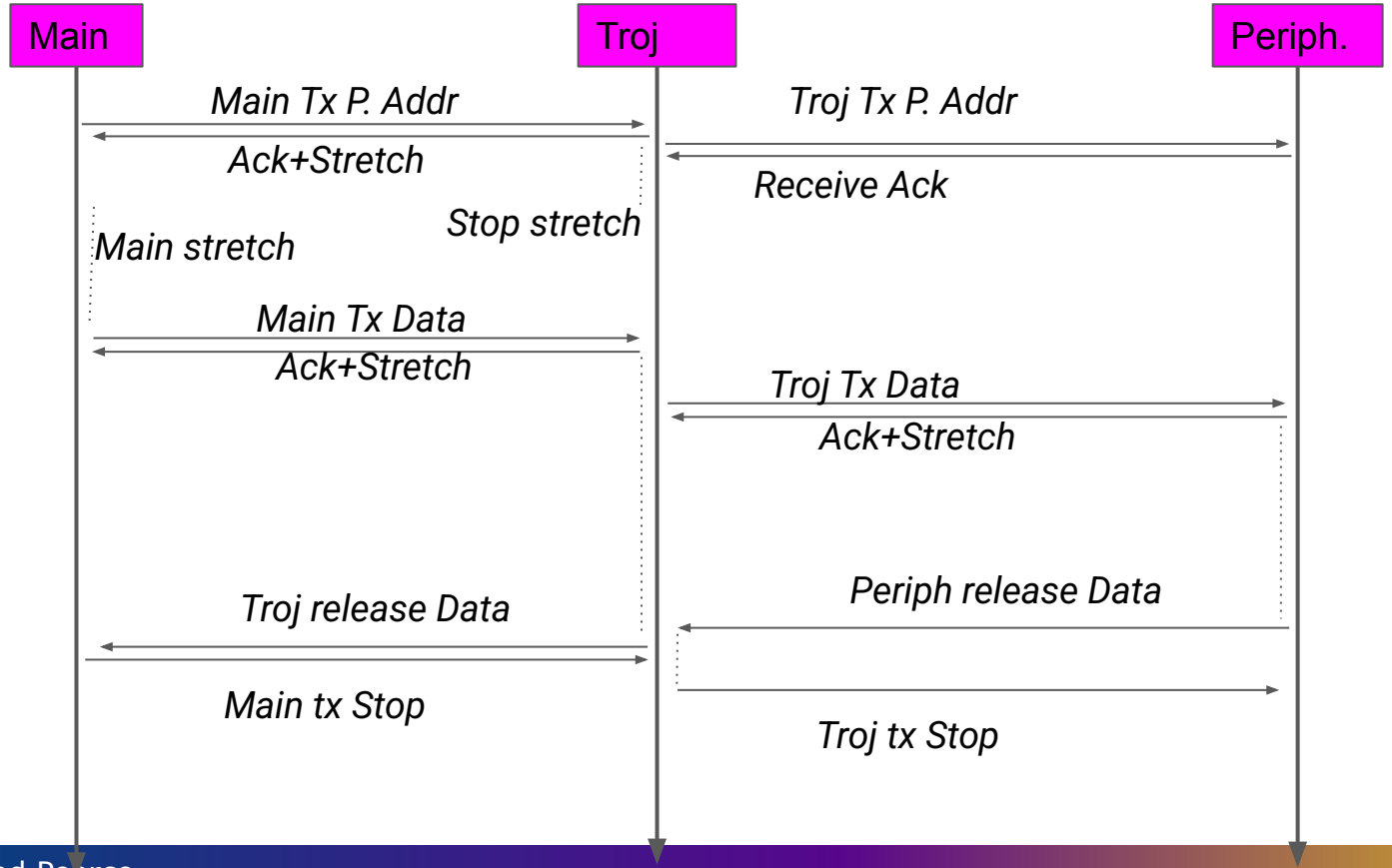
Process for a “clock-stretch”



Process for a “clock-stretch” MitM attack



Process for a “clock-stretch” MitM attack



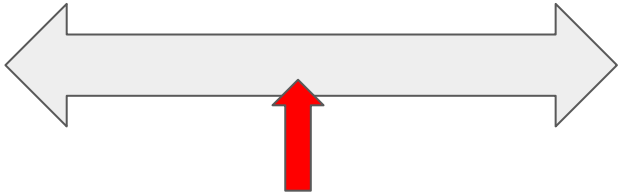
Scenario: AppMicro talks to EEPROM

Hello, what is value at 0x1234?

It is 5678

Scenario: AppMicro talks to EEPROM

Hello, what is value at 0x1234?

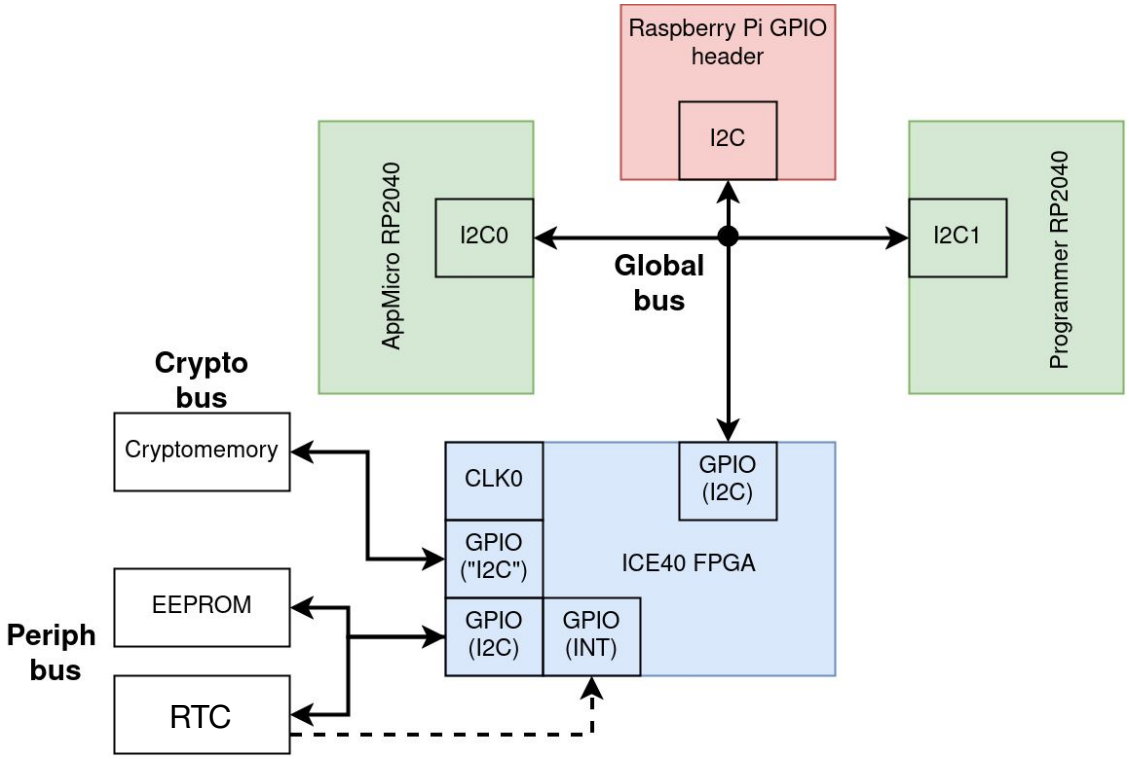


It is 9999

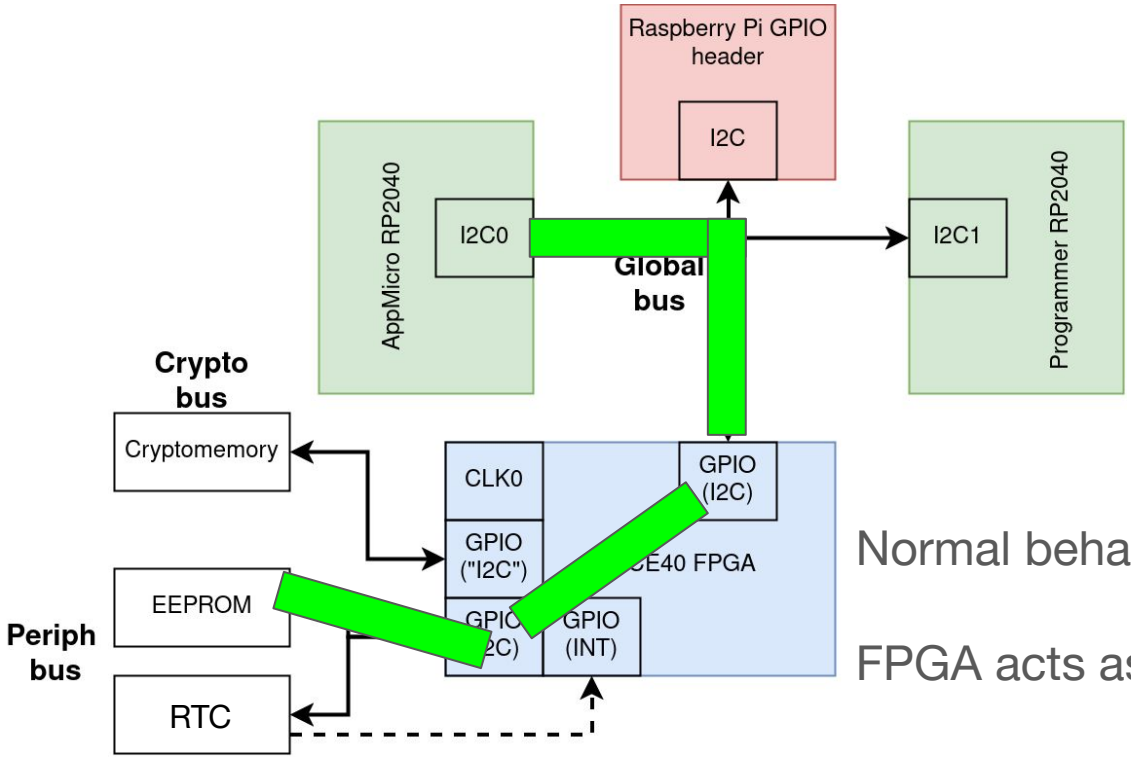
You could interfere!

You could implement a clock-stretching attack.

Hackster I2C Network



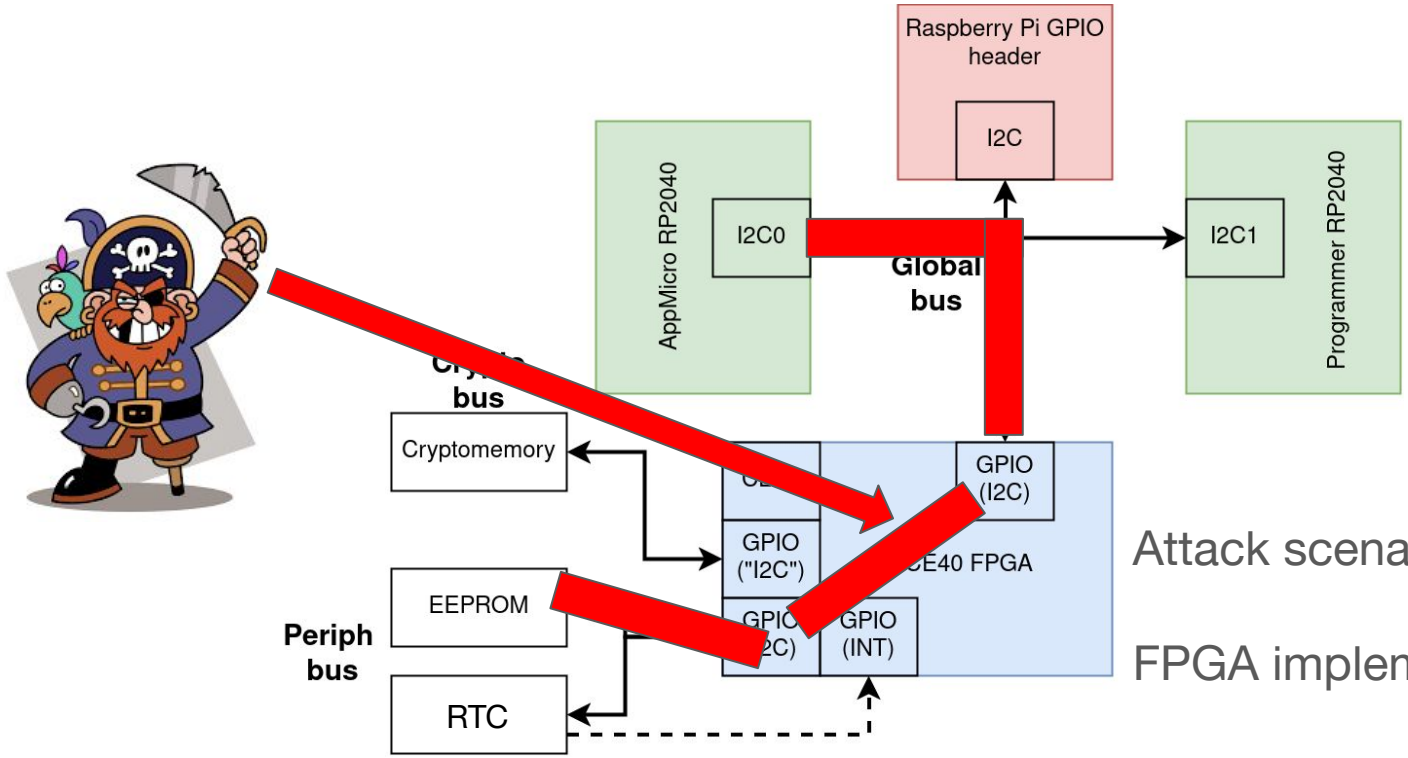
Hackster I2C Network



Normal behaviour

FPGA acts as “pass through”

Hackster I2C Network



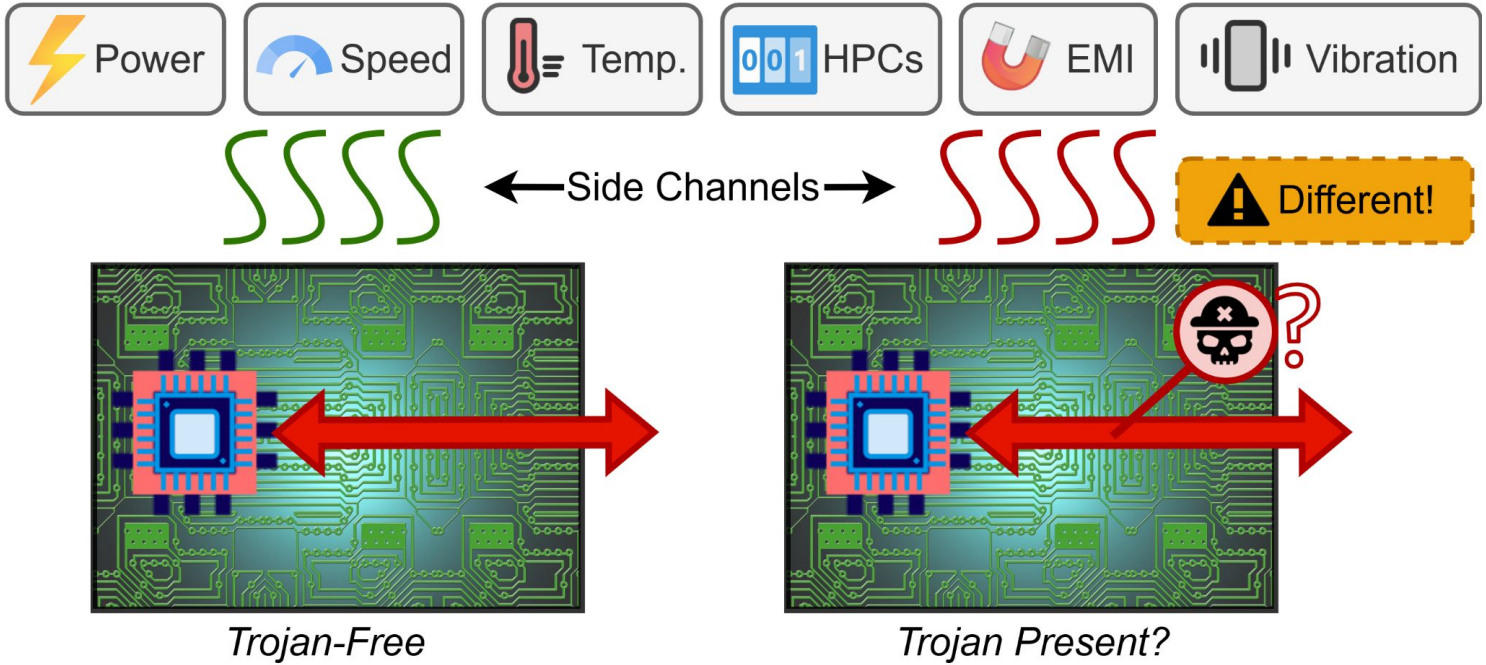
Attack scenario

FPGA implements I2C Trojan

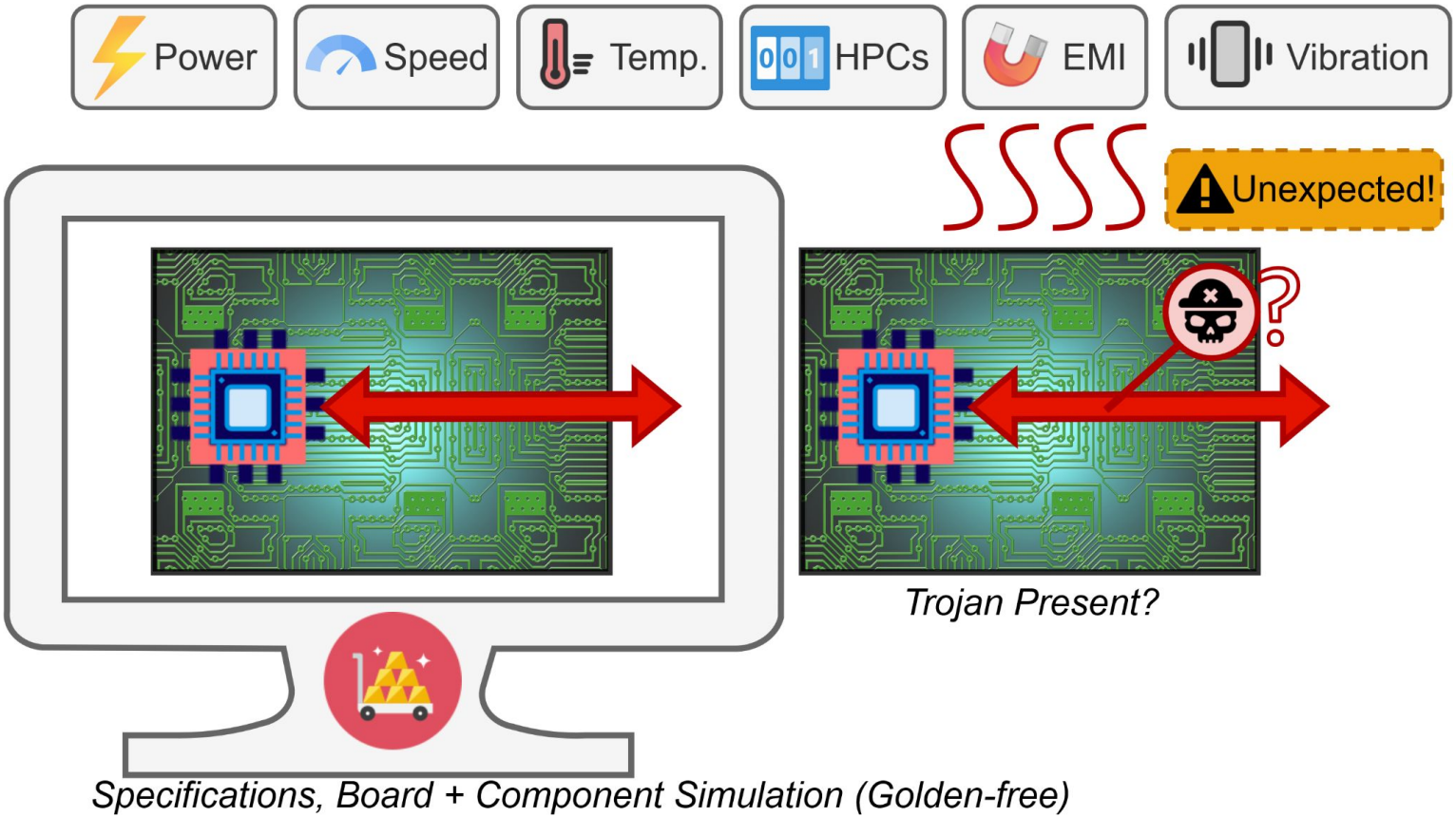
How to detect a hardware Trojan?

- Research focus has primarily been at IC/IP level, not PCB.
  - Many IC level detection strategies not directly applicable
- Static PCB-approaches
  - X-ray, imagery: impose constraints on handling and production
- Dynamic approaches
  - Our focus: run-time operational tests.
  - Side channel comparison against golden model.
  - Not yet demonstrated as possible for PCB level HW trojans.

Detection principle - side channels!

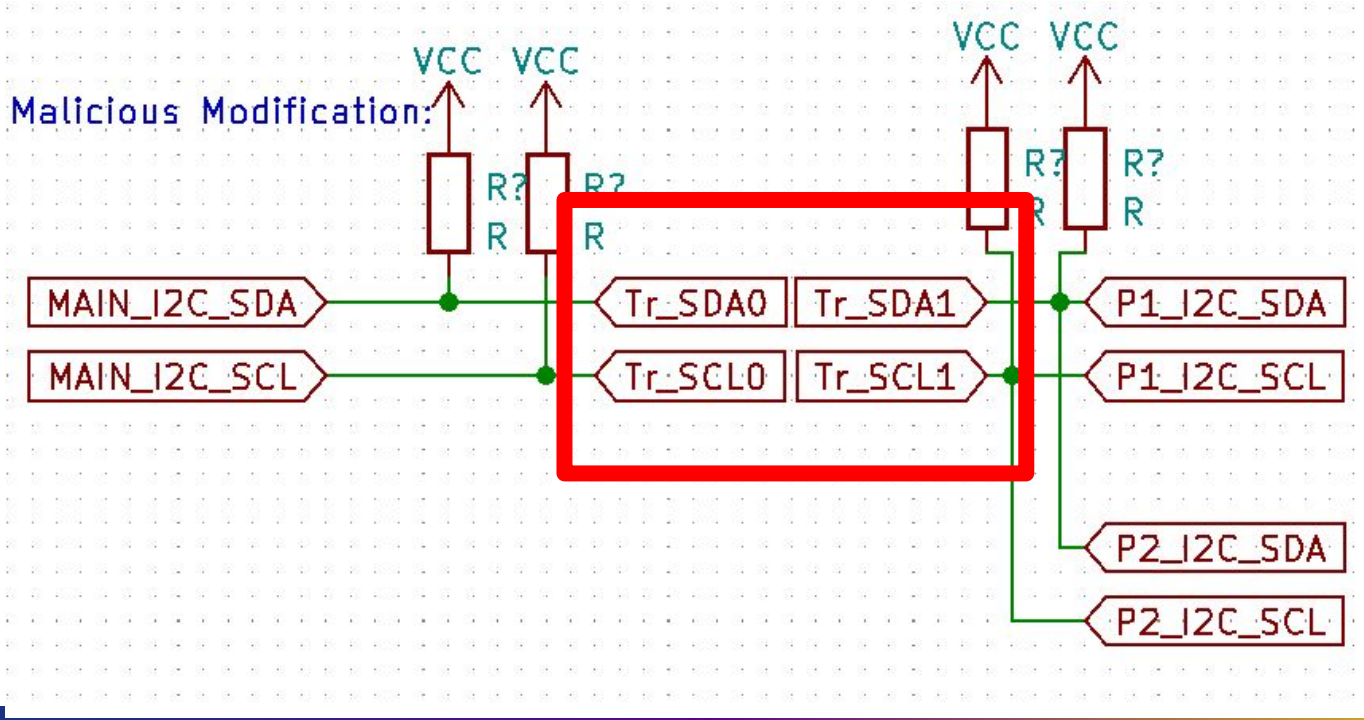


Detection principle - side channels!



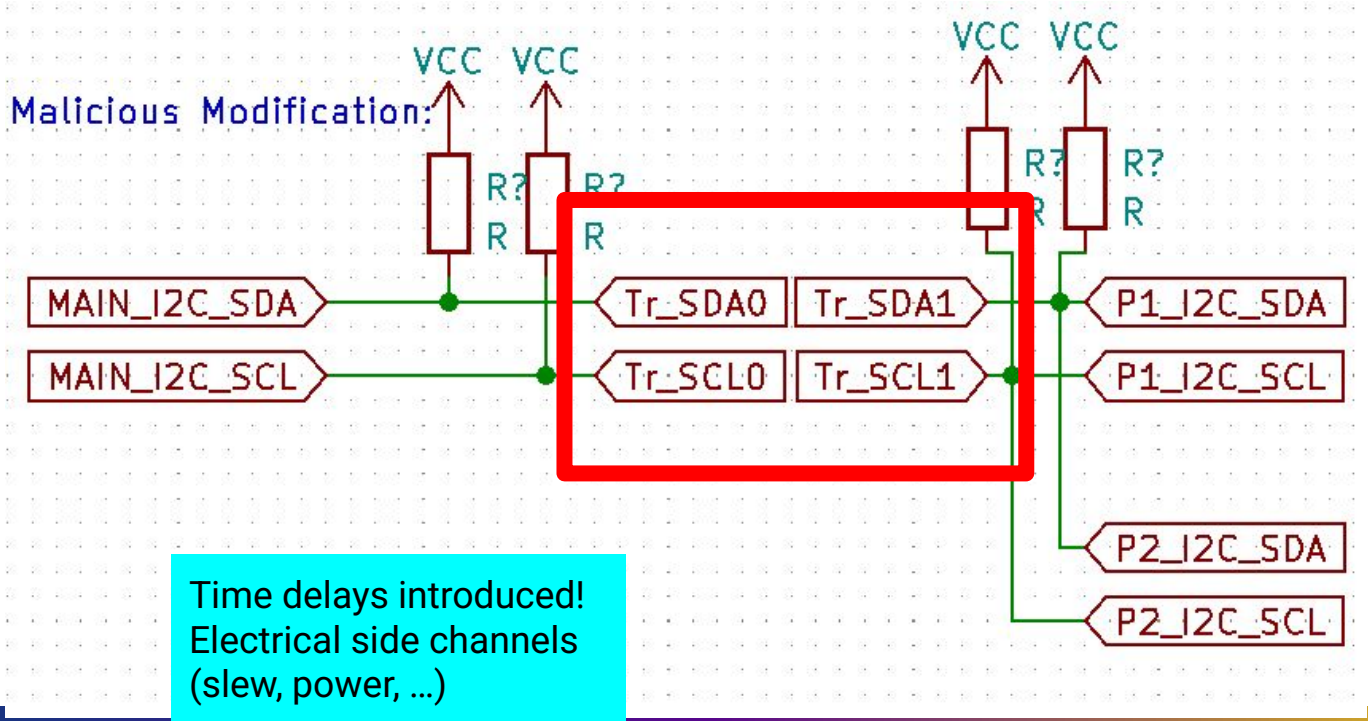
Detecting these Trojans?

- Ideas?

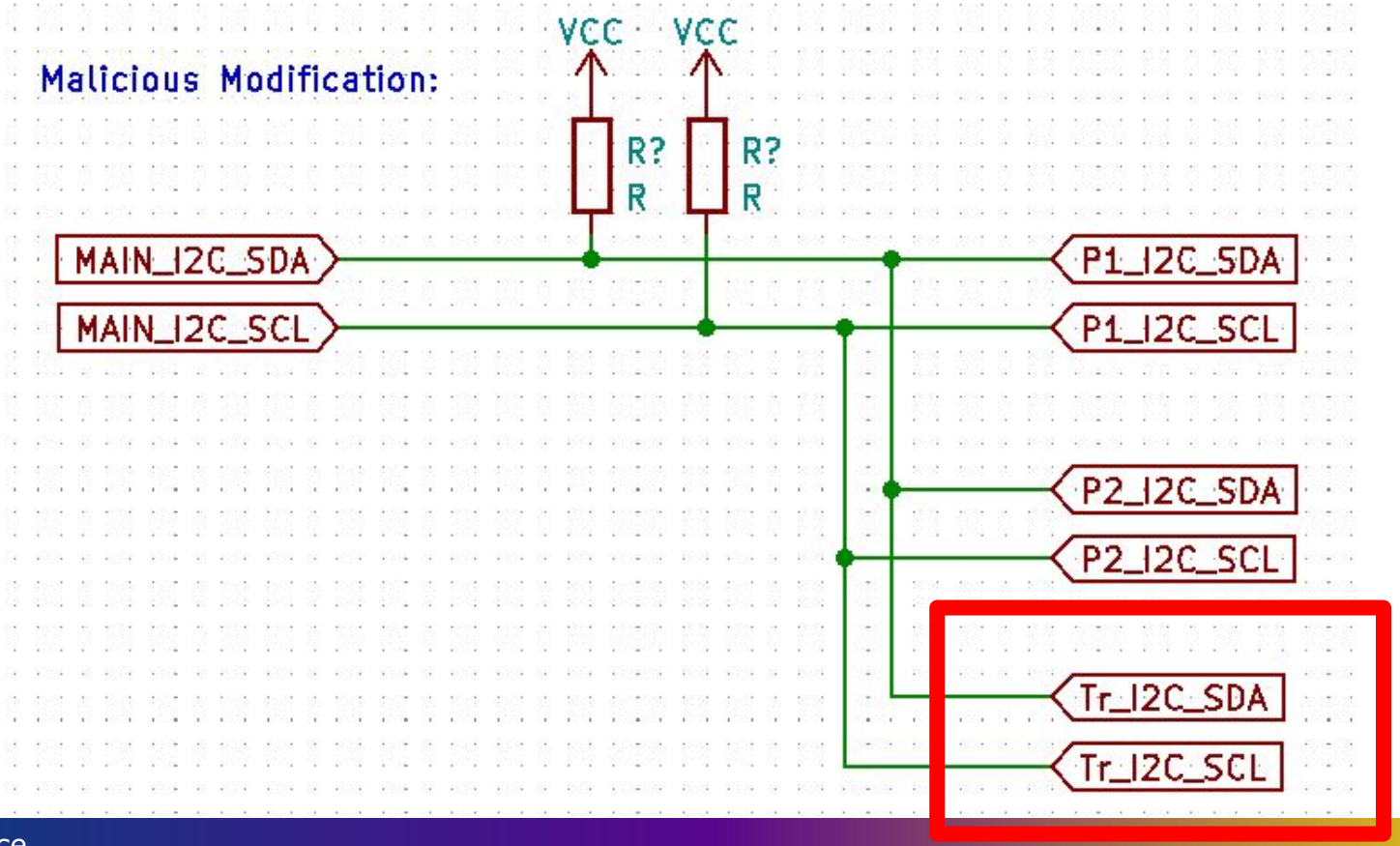


Detecting these Trojans?

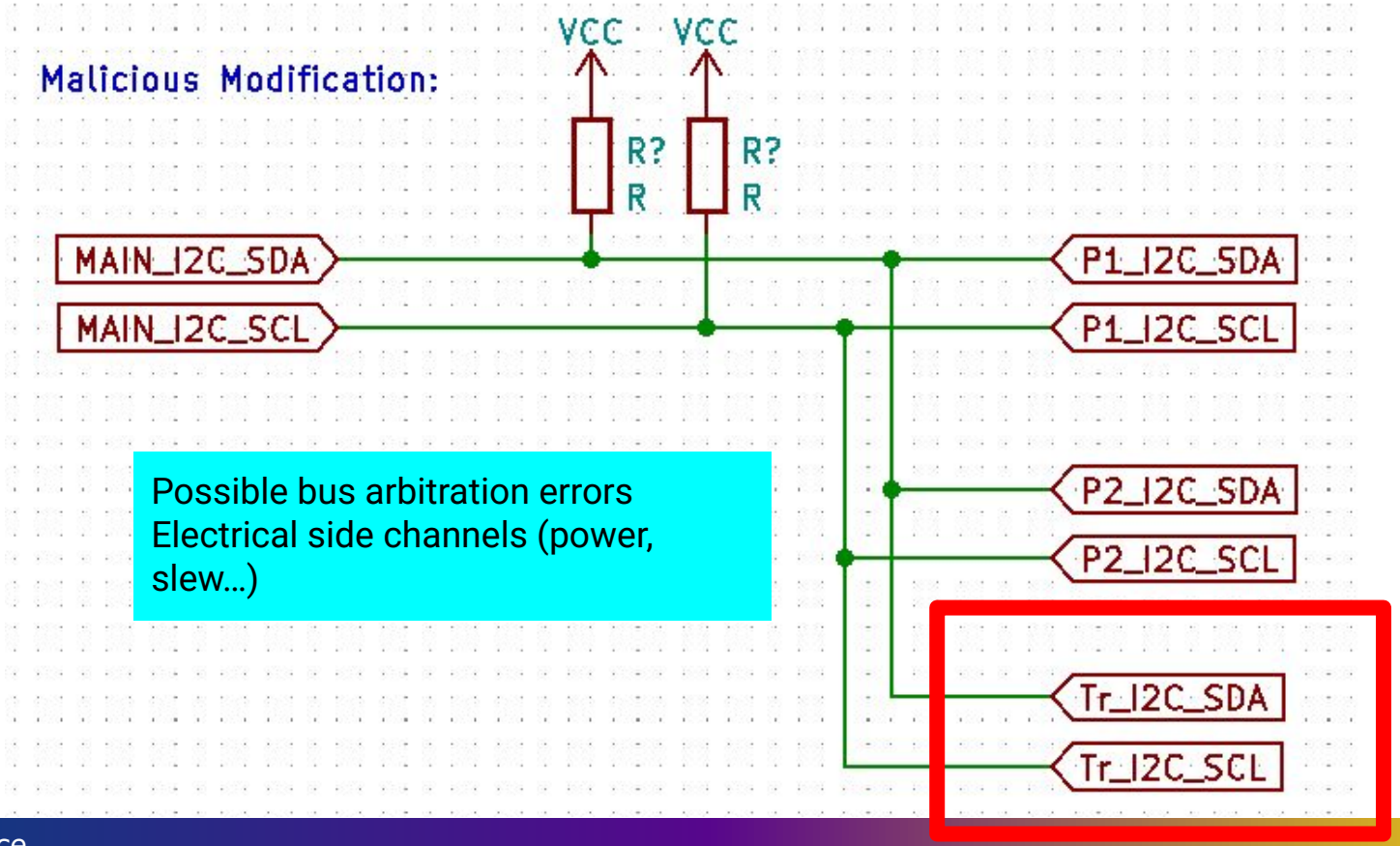
- Ideas?



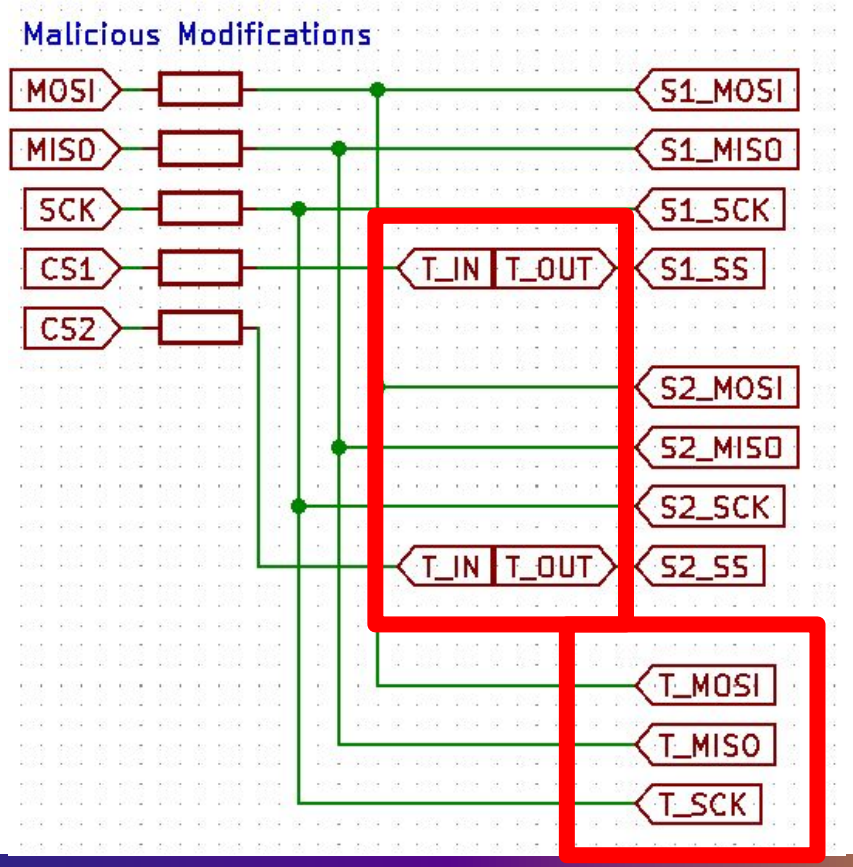
Detect this?


Malicious Modification:
VCC
VCC
R?
R
R?
R
MAIN_I2C_SDA
MAIN_I2C_SCL
P1_I2C_SDA
P1_I2C_SCL
P2_I2C_SDA
P2_I2C_SCL
Tr_I2C_SDA
Tr_I2C_SCL

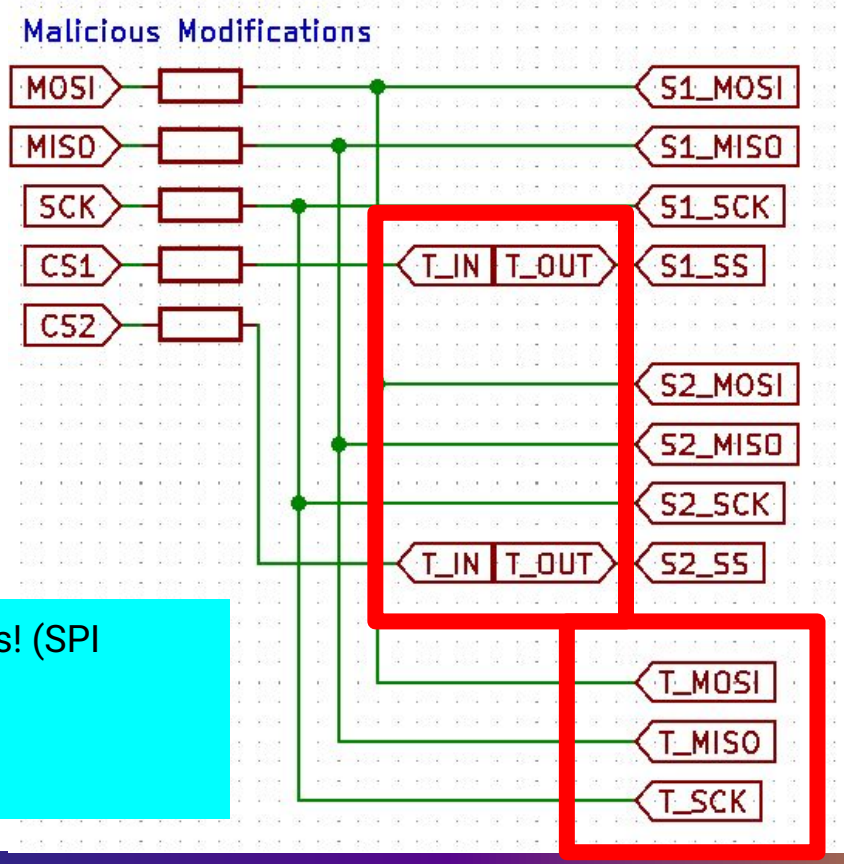
Detect this?


Malicious Modification:
VCC
VCC
R?
R
R?
R
MAIN_I2C_SDA
MAIN_I2C_SCL
P1_I2C_SDA
P1_I2C_SCL
P2_I2C_SDA
P2_I2C_SCL
Tr_I2C_SDA
Tr_I2C_SCL
Possible bus arbitration errors
Electrical side channels (power, slew…)

Detect this?

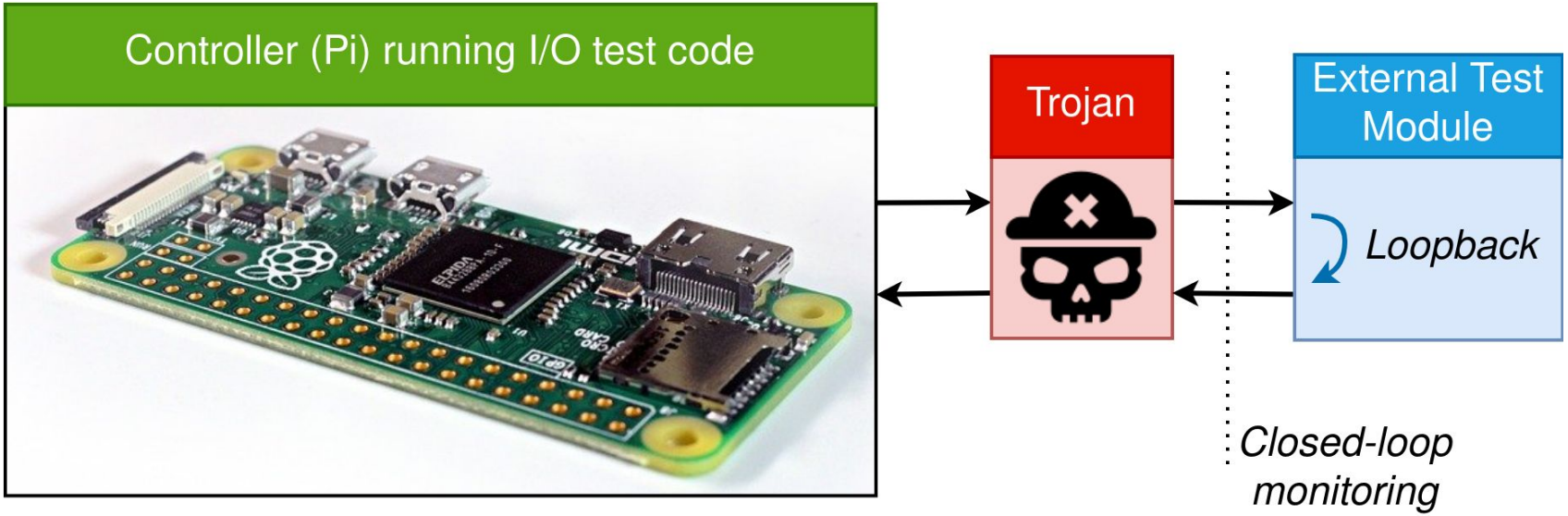

Malicious Modifications
MOSI
MISO
SCK
CS1
CS2
S1_MOSI
S1_MISO
S1_SCK
T_IN
T_OUT
S1_SS
S2_MOSI
S2_MISO
S2_SCK
T_IN
T_OUT
S2_SS
T_MOSI
T_MISO
T_SCK

Detect this?



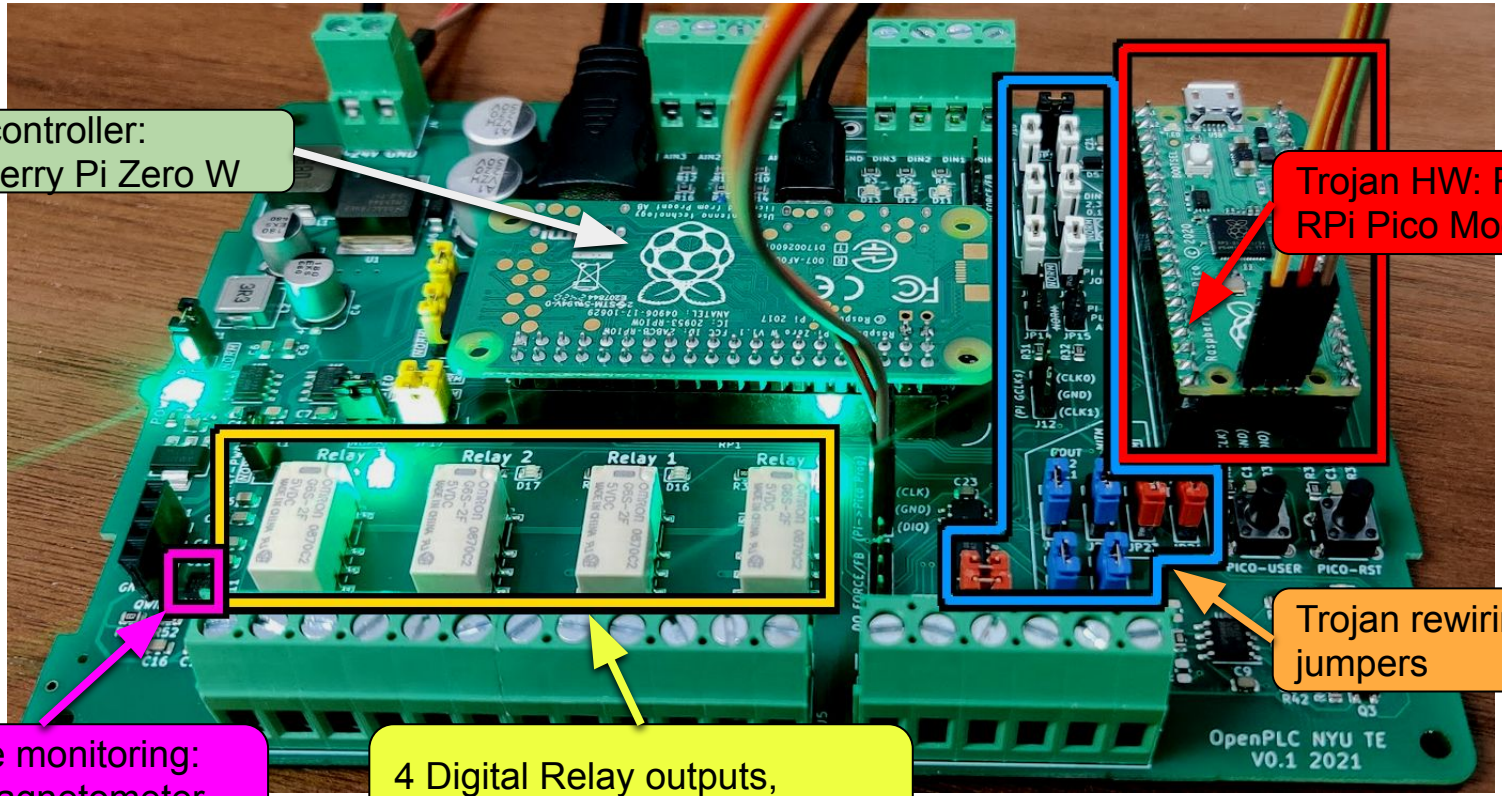
No protocol timing delays! (SPI doesn't allow)
Possible bus errors
Electrical side channels

Closed-loop Trojan Detection

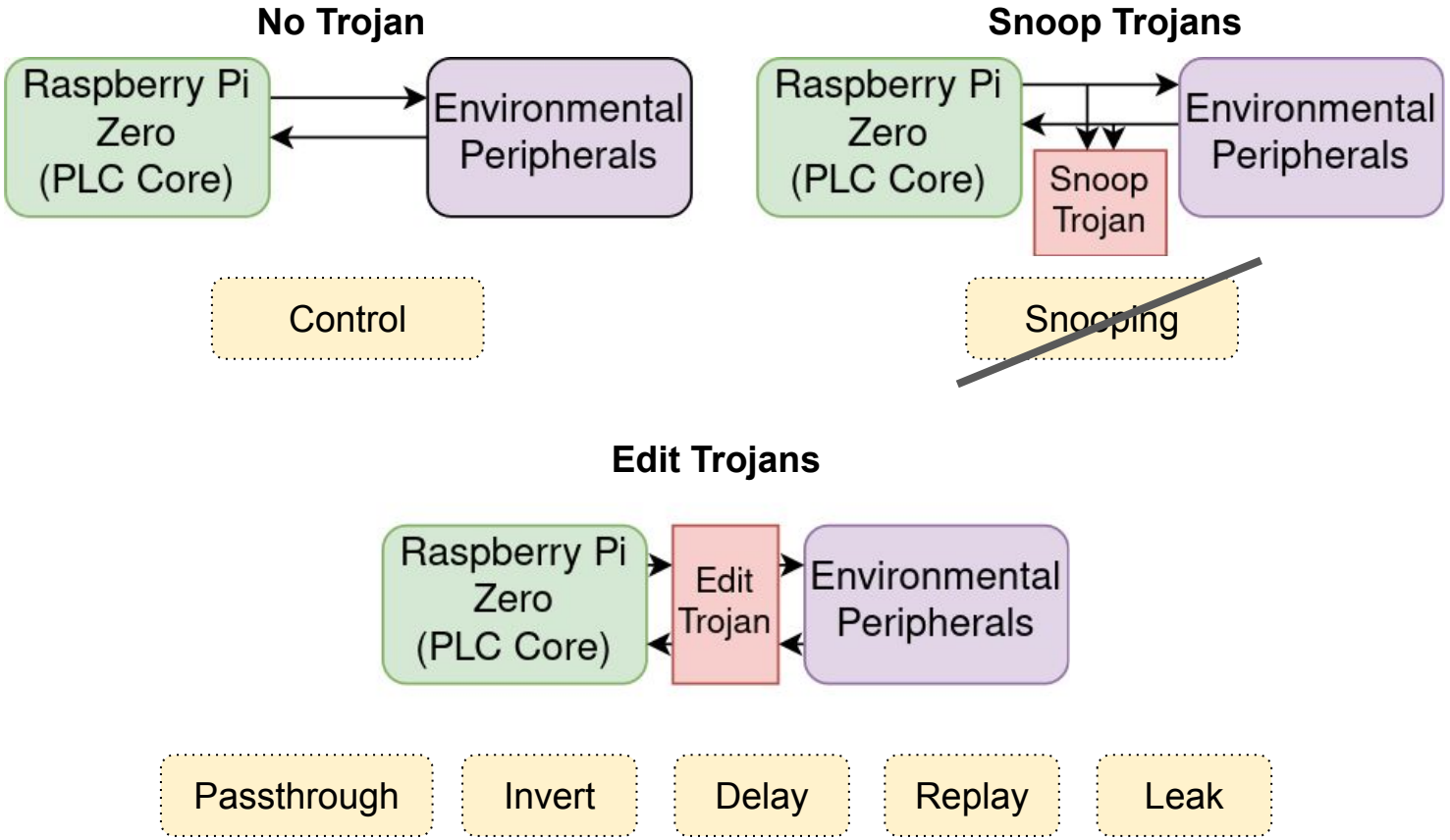


- Software-driven excitations - fuzzing
- Analyze all I/O and side-channels
- Simplest possible instance: direct connection (e.g. for GPIO)

NYU OpenPLC “Trojan Edition”

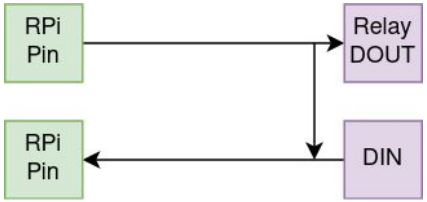


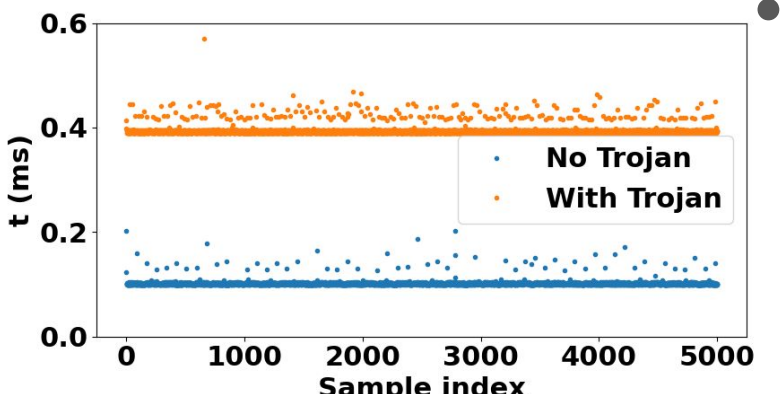
What Trojans can be detected?



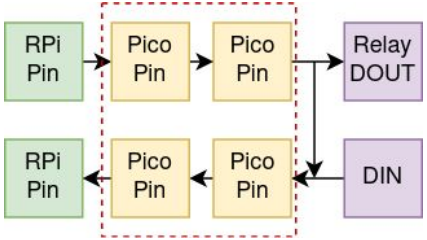
Example: GPIO MitM Trojan

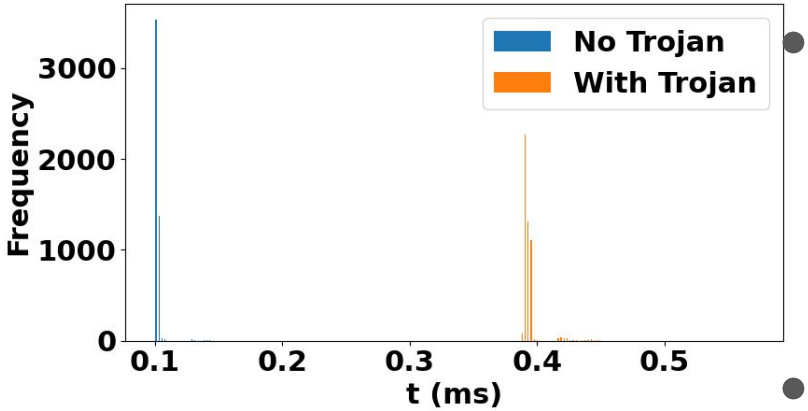
Expected: Short delay





Trojan: Adds delay

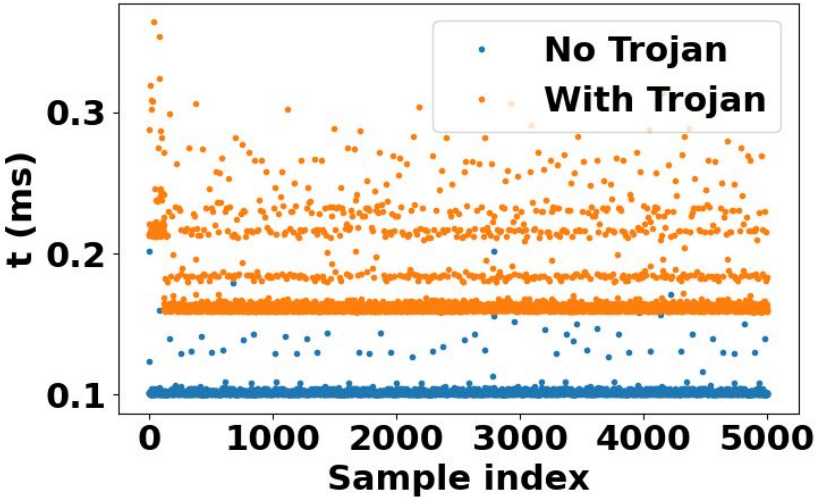


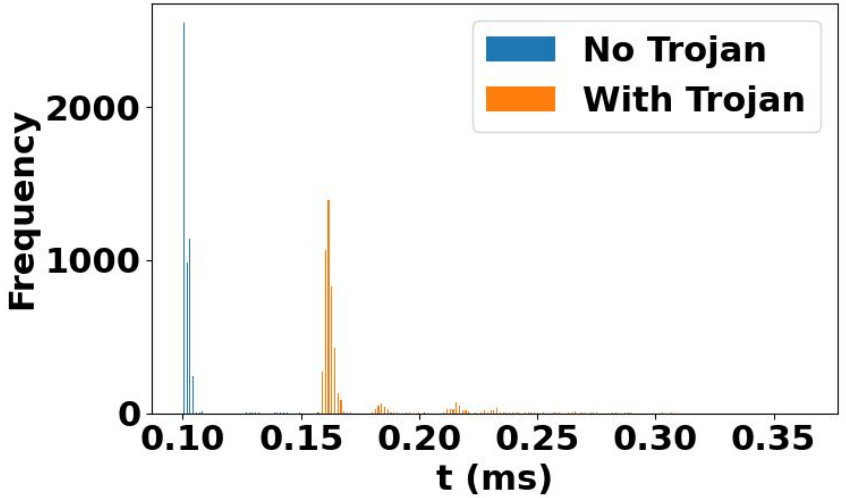


- Straightforward detection:
  - 500 loop-back cycles
  - Light-weight trojan identified
    - 250MHz
- *Golden-free* check:
  - OS syscall ~220ns for GPIO,
    Trojan adds ~600ns per cycle
- What about faster HW?

FPGA-based edit Trojan on GPIO

- FPGA “faster” than microcontroller-based (combinational logic)
  - Still has some delay - input, output buffers, analog protection…

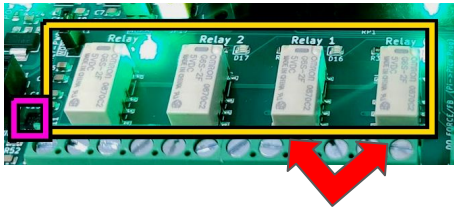


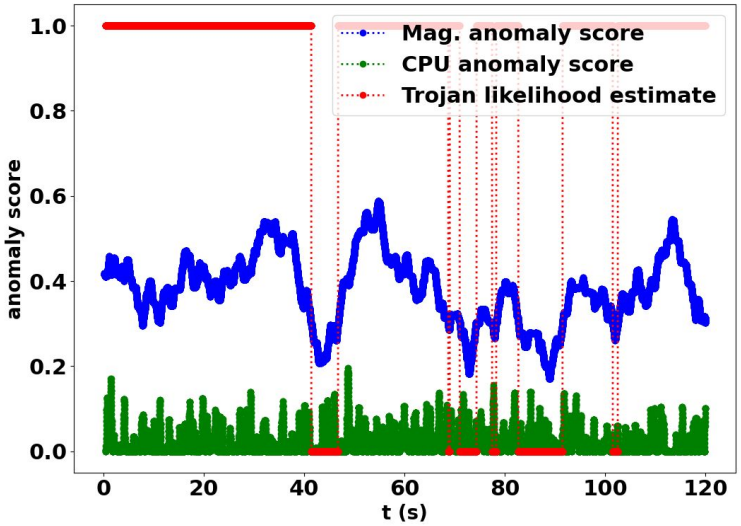


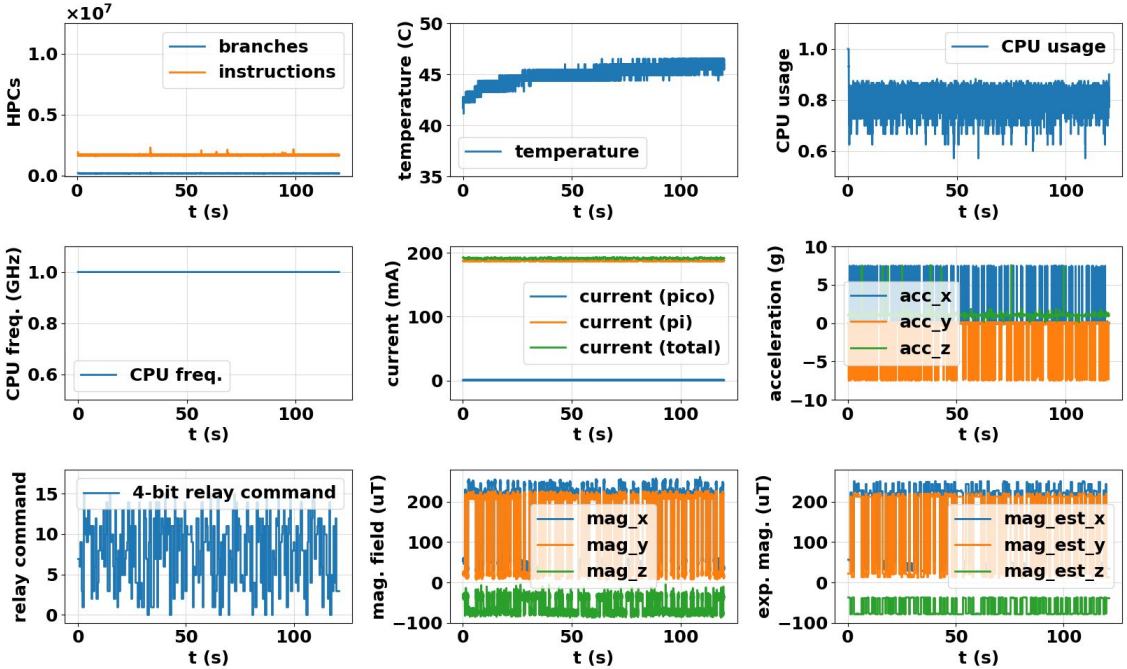
- Extra delay - one additional I/O system call?

Anomaly-based side channel detection

Invert Trojan







Side channel data combined with ML:
Random Forest Classifiers

Discussion

Trojan	Accuracy	Precision	Recall	F1-Score
Invert Trojan	0.877	0.911	0.836	0.872
Delay Trojan	0.959	0.925	1.000	0.961
Replay Trojan	0.959	0.925	1.000	0.961
Acoustic Info. Leakage Trojan	0.943	0.922	0.968	0.945

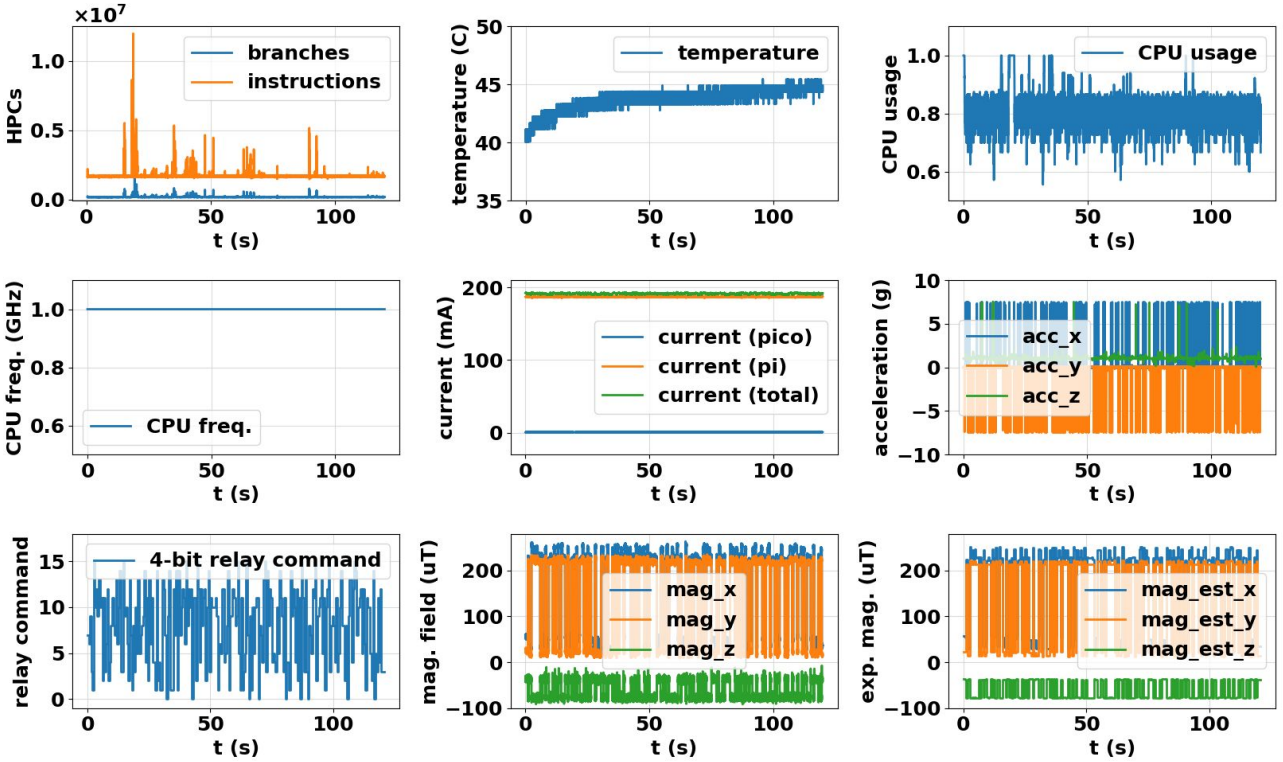
- **All Edit Trojans detected** - side channel loop-back detection viable for HW Trojans
- New open-source test-bed framework for HW Trojans
  - ‘Trojan-in-the-loop’
- Comparing with ‘golden’ models helpful but not necessary
  - Design information (specifications), simulator models

❑ **H. Pearce,** V. R. Surabhi, P. Krishnamurthy, J. Trujillo, R. Karri, and F. Khorrami, “Detecting Hardware Trojans in PCBs Using Side Channel Loopbacks”, *IEEE Transactions on Very Large Scale Integration Systems*, 2022.

Appendix - other measurements

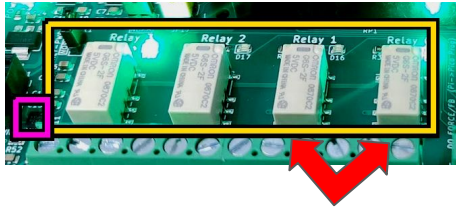
Anomaly-based side channel detection

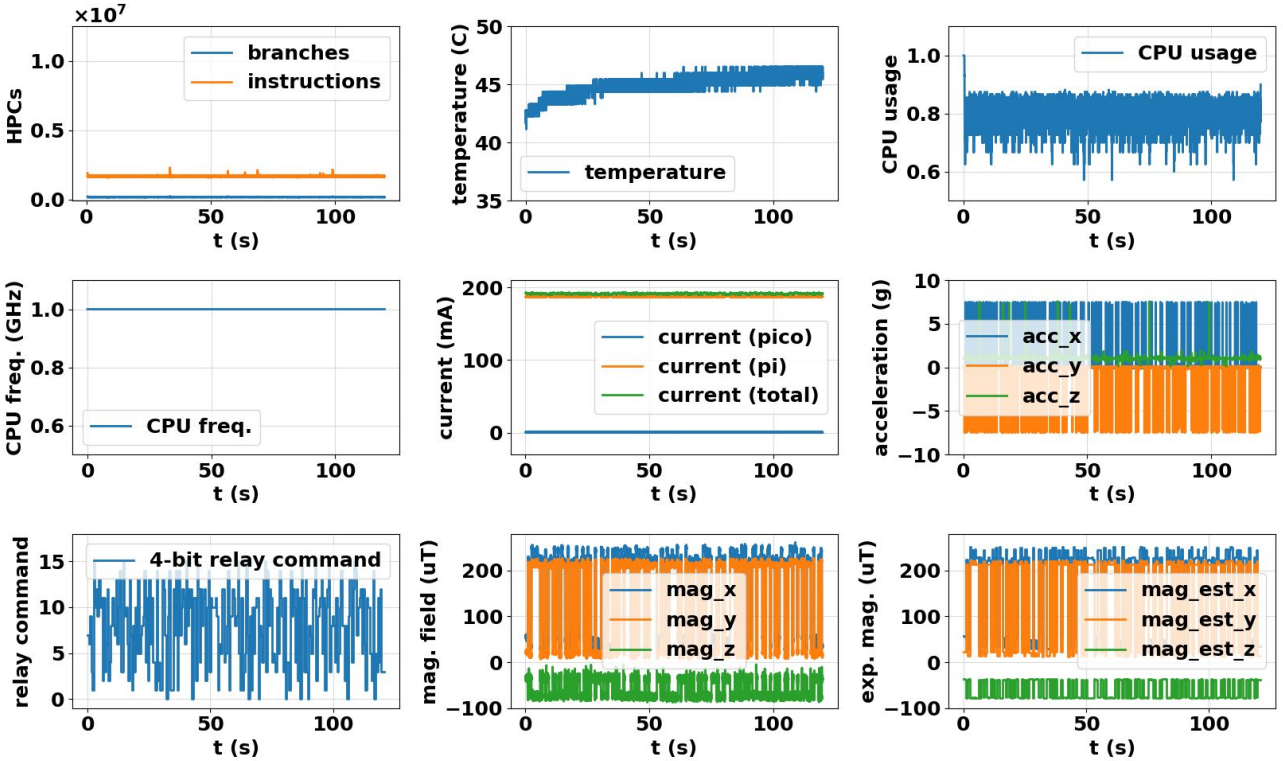
Control - No Trojan



Anomaly-based side channel detection

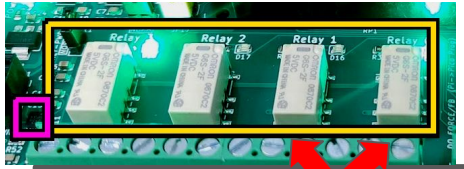
Invert Trojan

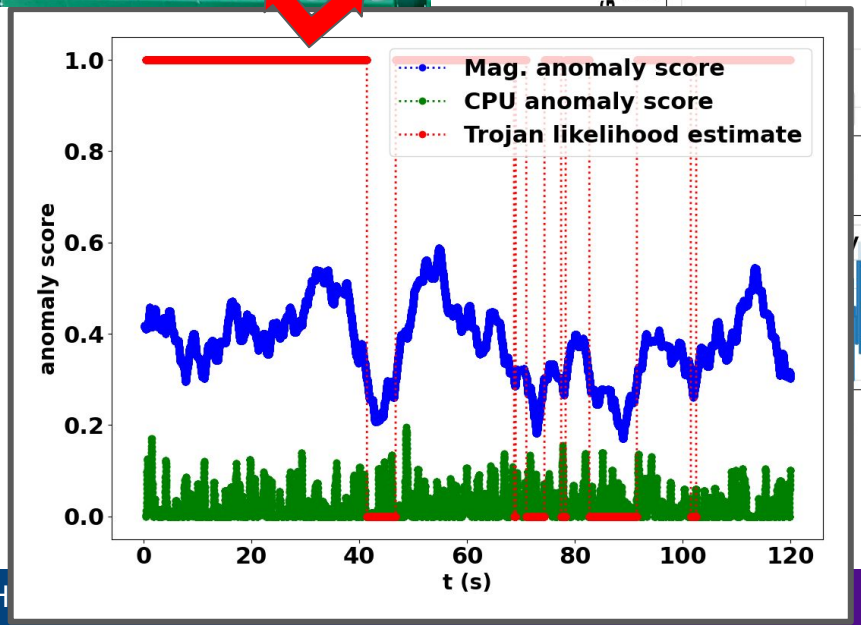


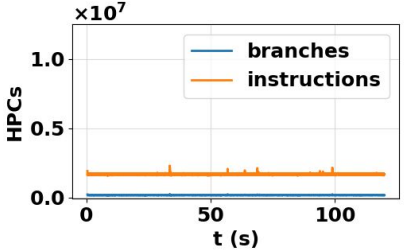


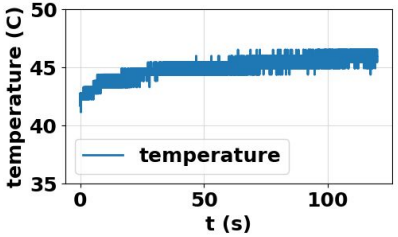
Anomaly-based side channel detection

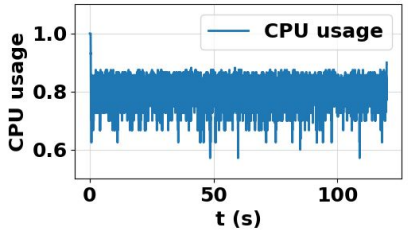
Invert Trojan

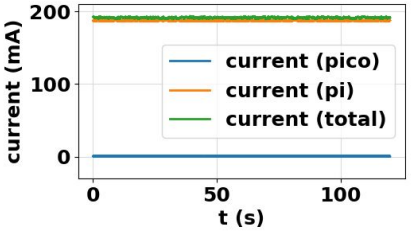


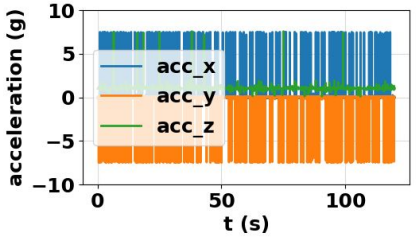


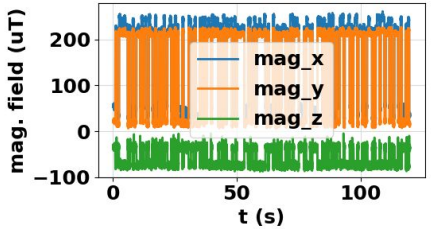


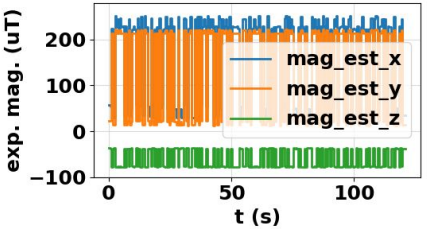






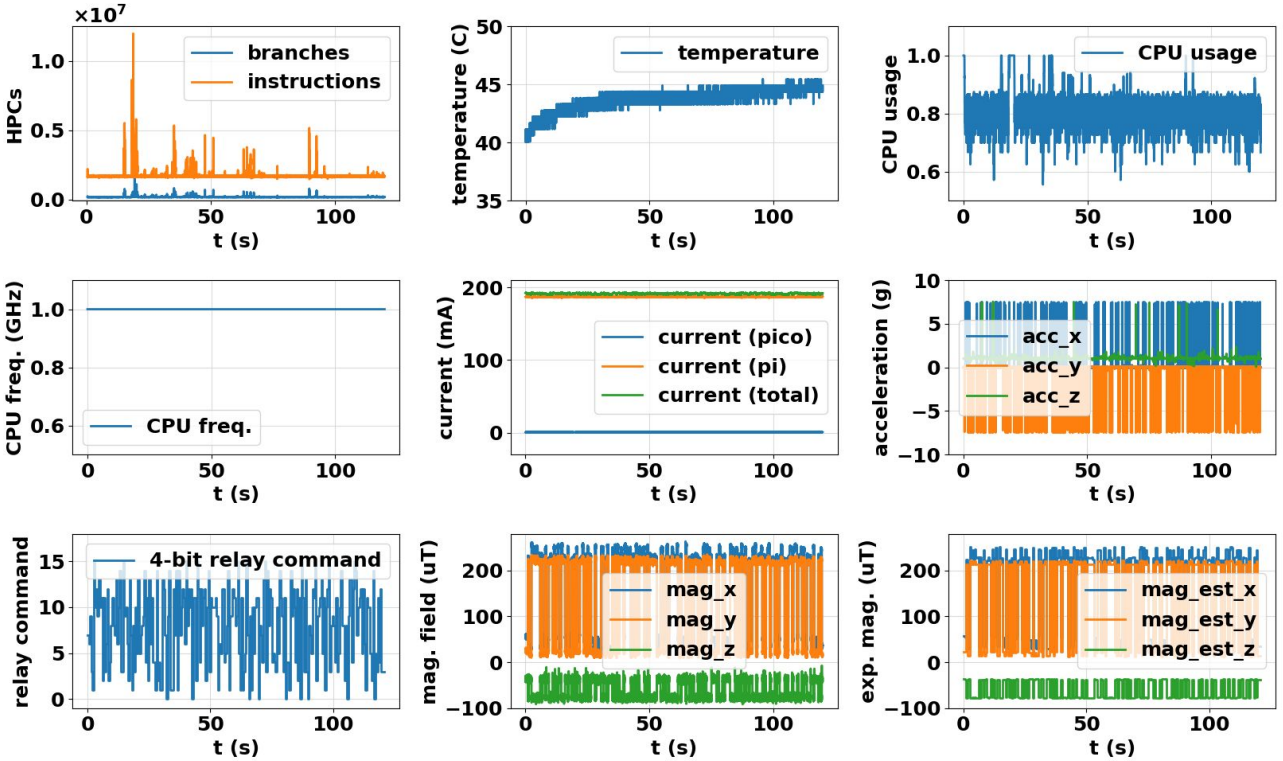






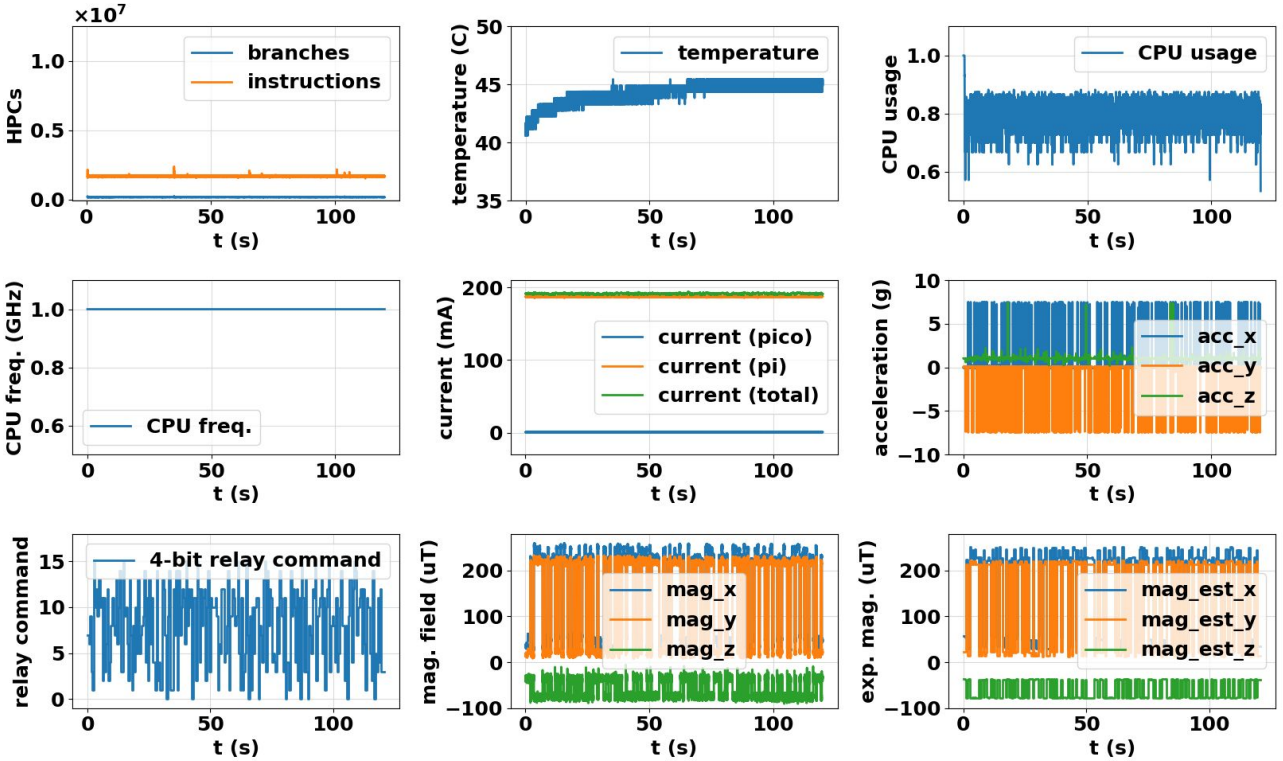
Anomaly-based side channel detection

Control - No Trojan



Other side channels

Delay Trojan



Other side channels

Delay Trojan