




Taking DMA Attacks to the Next Level:

How to do arbitrary reads/writes in a live and unmodified system using a rogue memory controller

Anna Trikalinou and Dan Lake

© 2017 Intel Corporation  #BHUSA / @BLACKHATEVENTS

Legal Notices and Disclaimers

- o This presentation contains the general insights and opinions of Anna Trikalinou and Dan Lake (collectively “Presenters”). The views, opinions, findings, and conclusions or recommendations expressed in this presentation are strictly those of the Presenters and do not necessarily reflect the official policy or position of the Presenters’ employer. The information in this presentation is provided for information only and is not to be relied upon for any purpose other than educational. Use at your own risk! Presenters make no representations or warranties regarding the accuracy or completeness of the information in this presentation. Presenters accept no duty to update this presentation based on more current information. Presenters are not liable for any damages, direct or indirect, consequential or otherwise, that may arise, directly or indirectly, from the use or misuse of the information in this presentation.
- o No computer system can be absolutely secure.
- o No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- o Intel, the Intel logo and Intel Core are trademarks of Intel Corporation in the United States and other countries.
- o *Other names and brands may be claimed as the property of others.

Who are we

- Anna Trikalinou, PhD
Research Scientist
Intel Labs, Intel Corporation
Email: anna.trikalinou@intel.com
- Dan Lake
Systems Engineer
Intel Labs, Intel Corporation
Email: dan.lake@intel.com

About this talk

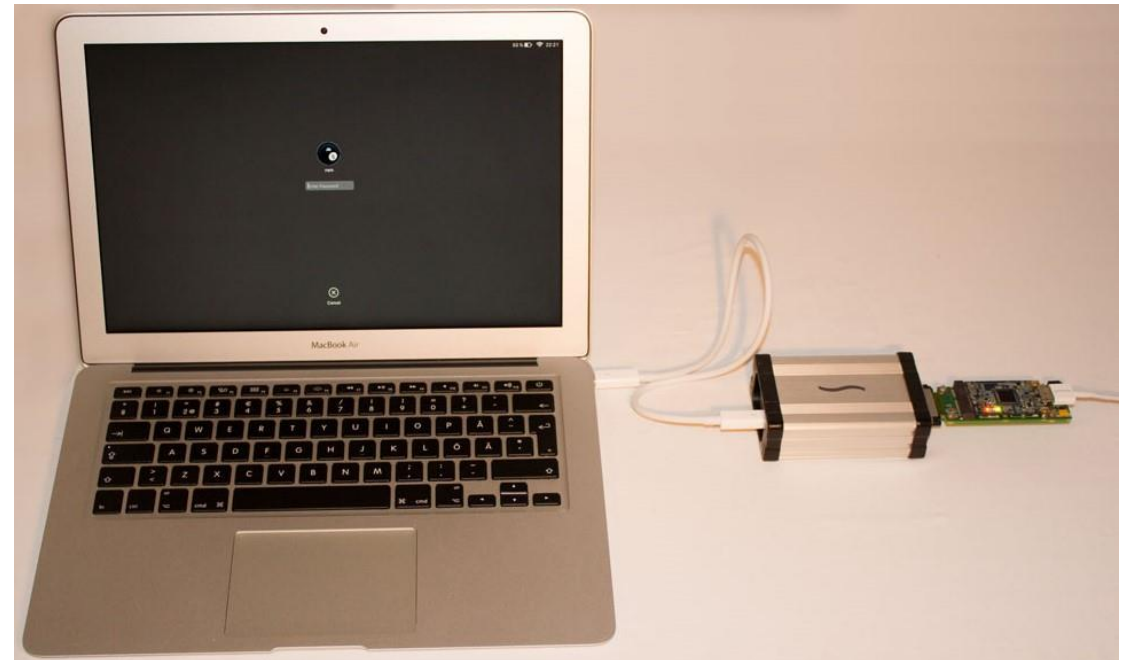
- This talk describes early research into a physical attack on DRAM memory
- We describe hardware design and protocol limitations
- If successful, the result is full access to all physical memory
- Applicable to all memories which follow the JEDEC spec and all architectures

Intro to DMA attacks

- Physical attack where the attacker connects to a DMA-capable port and gains full read/write access to the entire physical memory
 - e.g. PCI, PCI-E, FireWire, USB, etc.
- Goal:
 - Extract secrets (i.e. disk encryption keys)
 - Bypass platform's security policies (i.e. modify page tables)

Example of DMA attack

- Ulf Frisk demonstrated a DMA attack on a MacBook Air
- He was able to obtain the FileVault disk encryption password
- macOS Sierra 10.12.2 software update patched the security issue (Dec 2016)



Source: <http://blog.frizk.net/2016/12/filevault-password-retrieval.html>

Pros & Cons of Existing Attacks from Attacker's Perspective

- Pros:
 - HW, SW and tutorials are readily available
- Cons:
 - Require specific interface
 - Can be mitigated by BIOS & VT-d
 - Can be mitigated by blocking associated drivers and ports

Motivation

- Pros:
 - HW, SW and tutorials are readily available
- Cons:
 - Require specific interface
 - Can be mitigated by BIOS & VT-d
 - Can be mitigated by blocking associated drivers and ports

What if we could eliminate these?

DRAM design

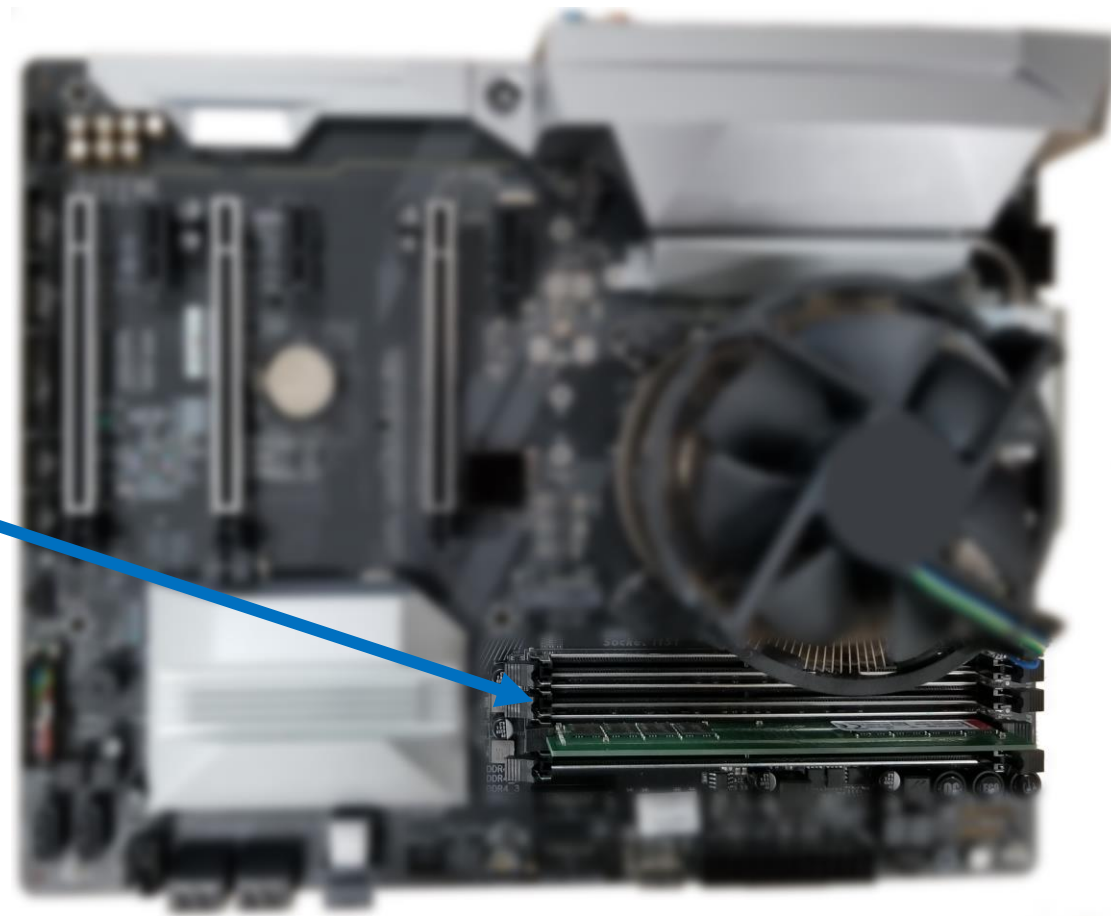
- A DIMM consists of a series of DRAM chips, mounted on a PCB.
- The DIMM is inserted into a DIMM socket on the motherboard, which then communicates with the processor.



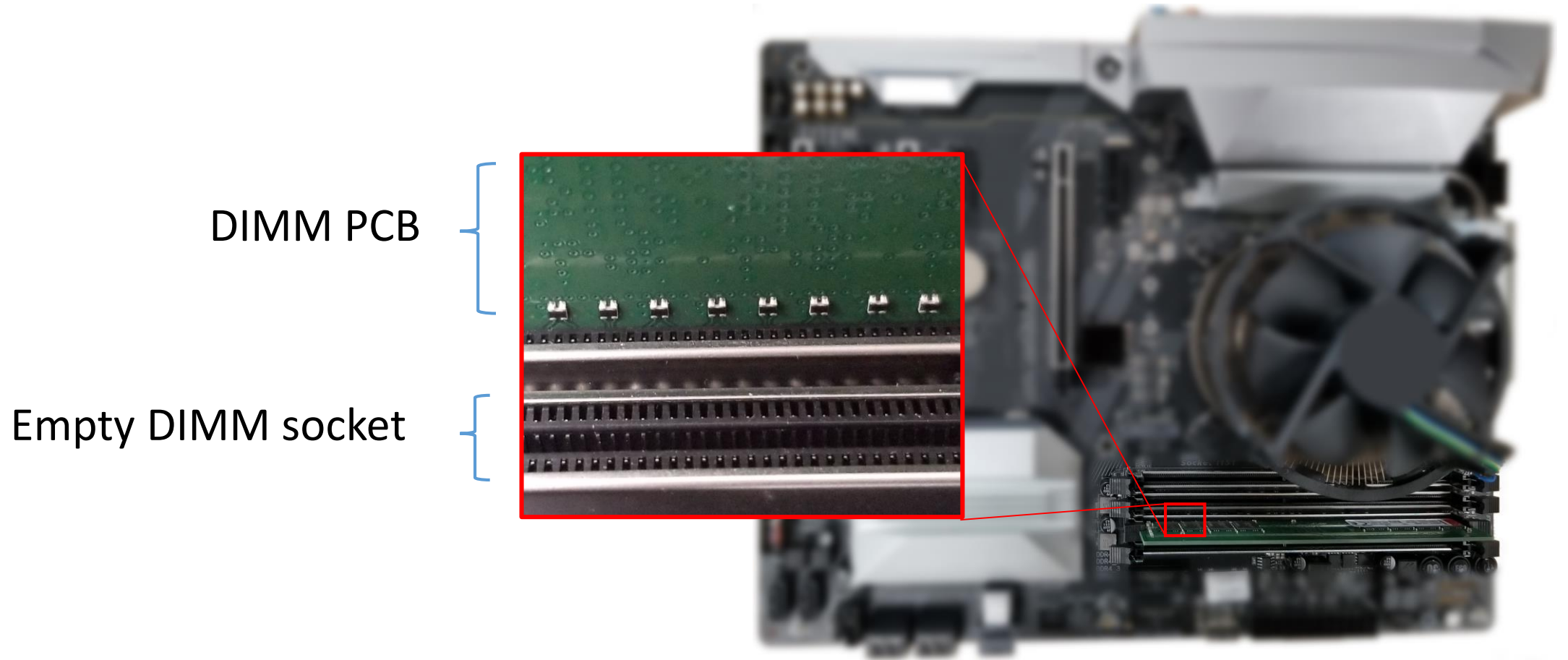
DRAM Placement

4 available DIMM Sockets
1 DIMM inserted

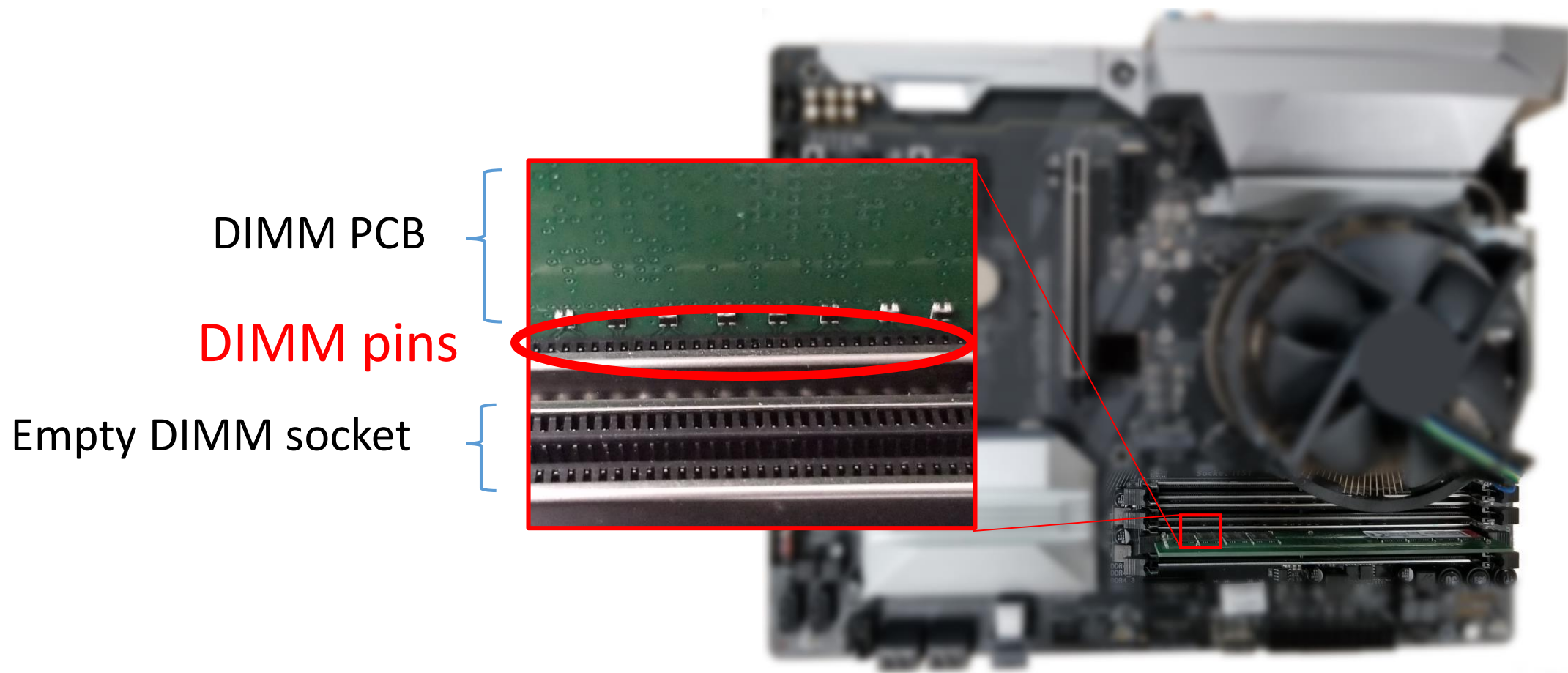
Reference System: Desktop with
current generation motherboard
and CPU



DRAM Placement

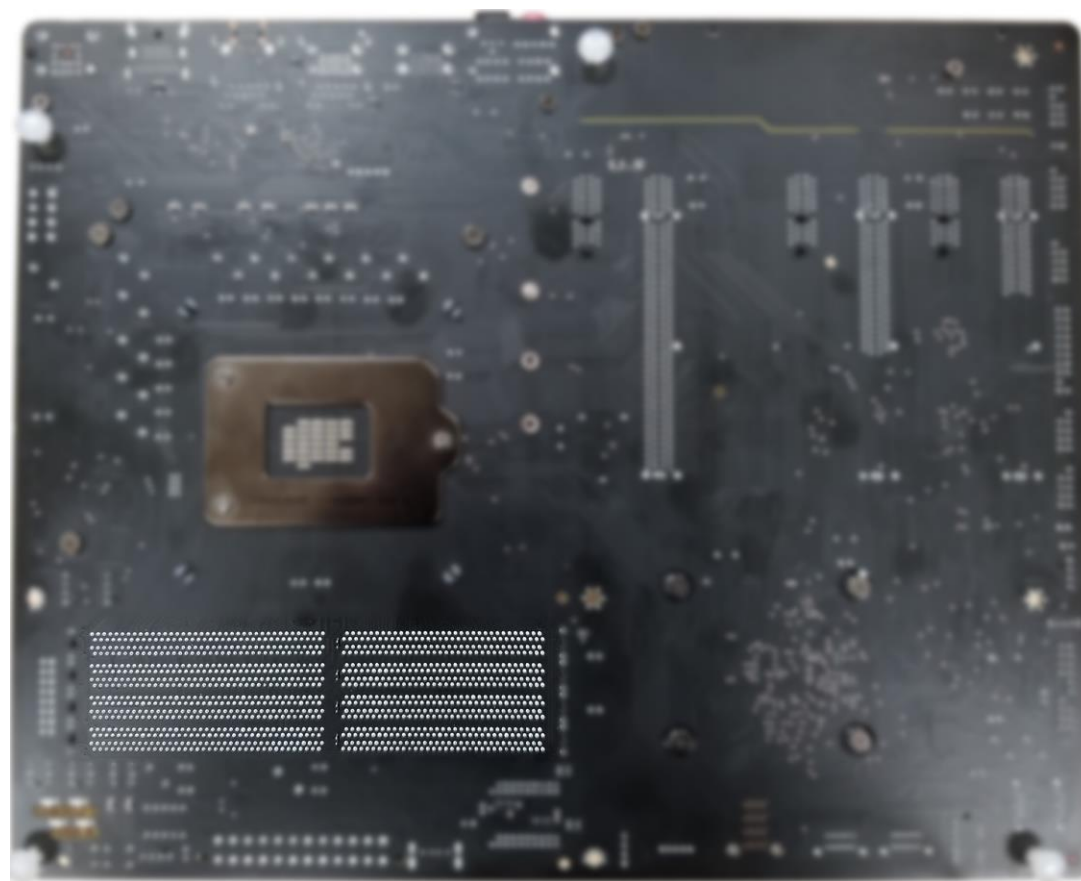


DRAM HW Design



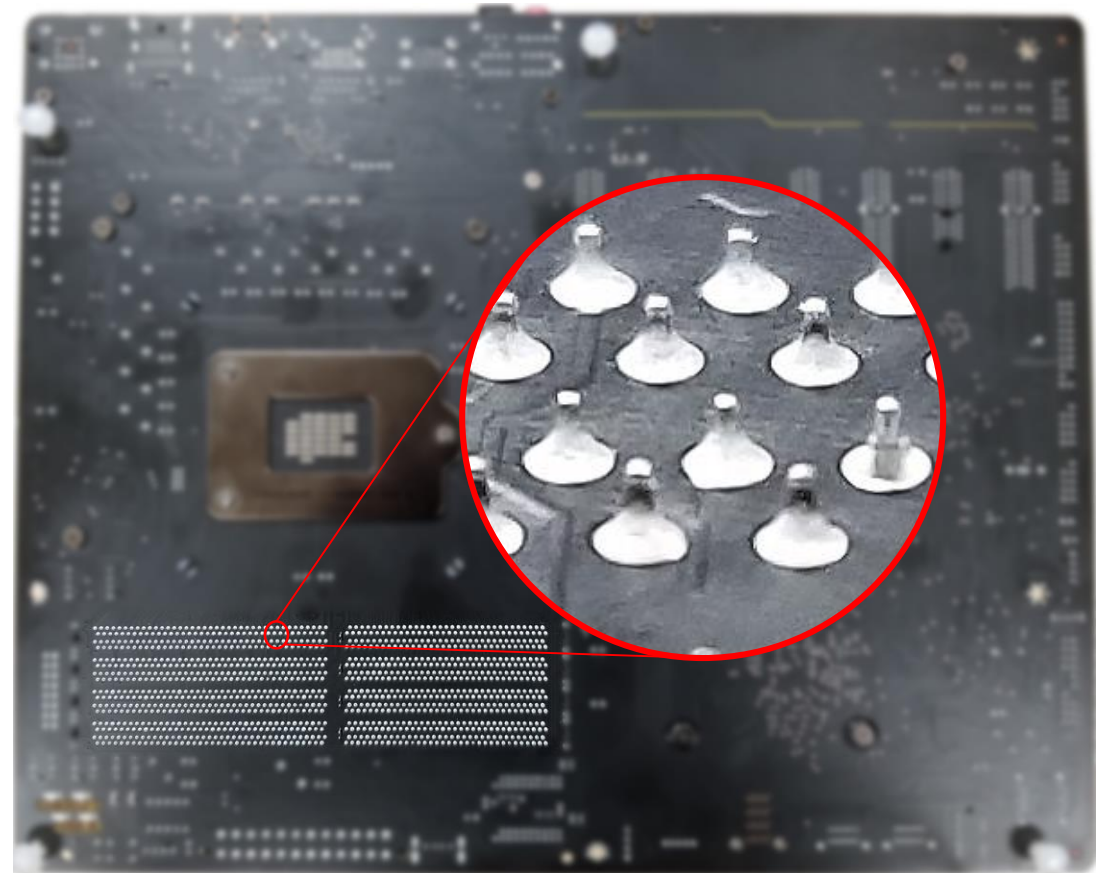
DRAM Placement on Back Side

Reference System: Desktop with current generation motherboard and CPU



DRAM HW Design

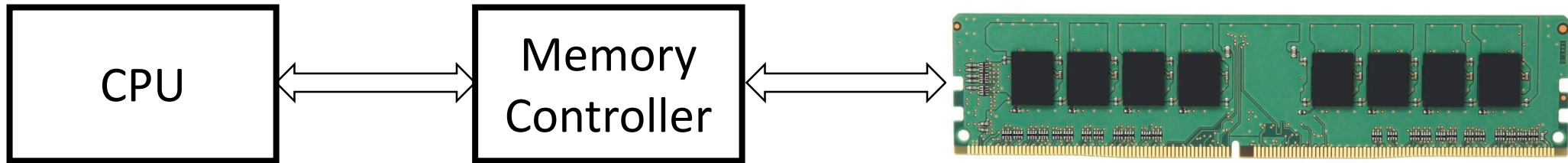
- All 288-pins of each DIMM socket are exposed on both sides of the motherboard
- Pins are electrically connected



How could we exploit that?

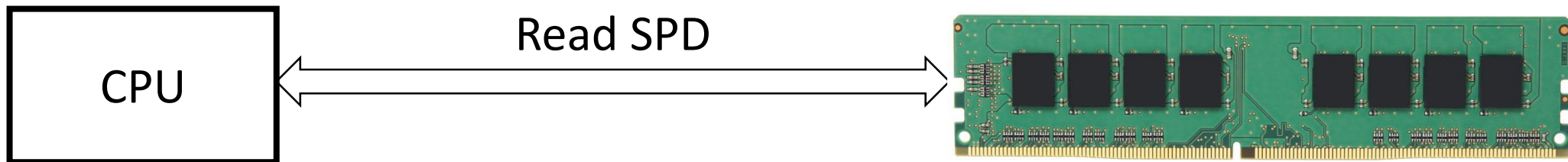
- What if we could plug into those pins and request reads/writes from the DIMM as a “memory controller”?

DRAM Functionality



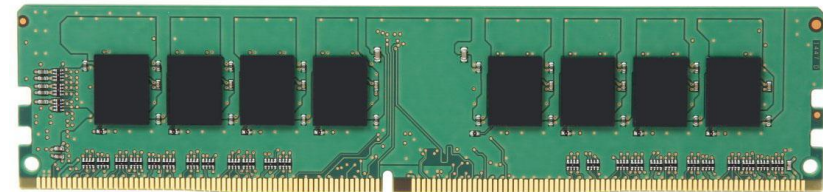
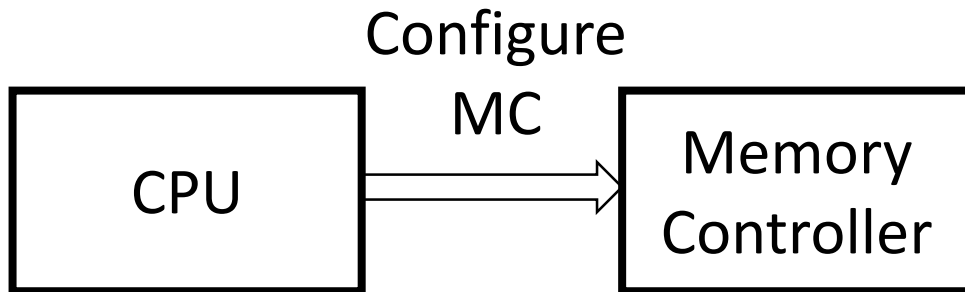
- Specified by JEDEC Standard
- Defines the set of requirements that must be satisfied by all memory modules and all architectures

DRAM Functionality: Initialization



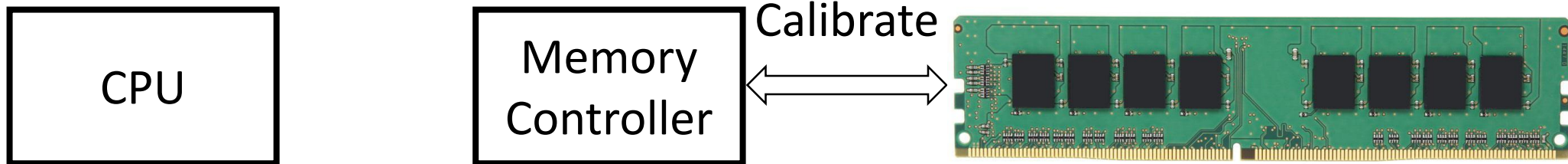
1. CPU reads DIMM's Serial Presence Detect (SPD) EEPROM data via SMBus
 - Manufacturer, module serial, supported voltage, minimum latencies, etc.
 - Bypass Memory Controller

DRAM Functionality: Initialization



2. CPU decides on a clock frequency and configures Memory Controller

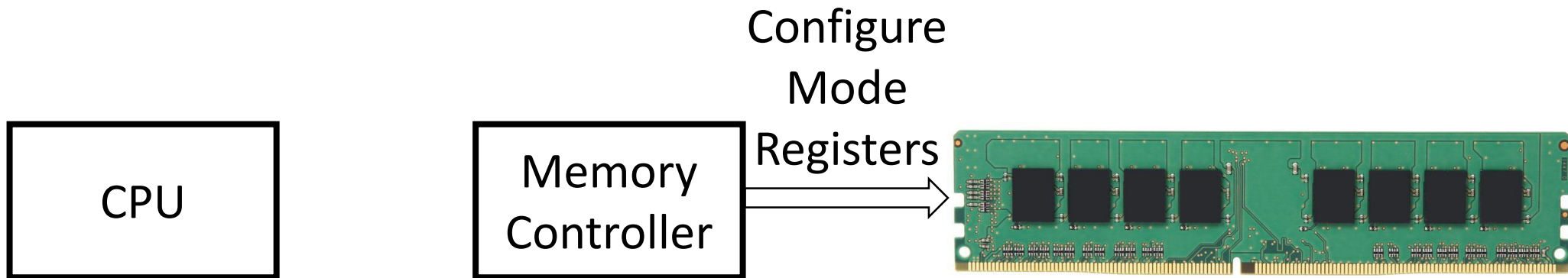
DRAM Functionality: Initialization



3. Memory Controller performs memory calibration

- MC is agnostic of motherboard and DIMM traces
- Calculates the round-trip time between MC and DIMM
- Calculates discrepancies between different traces

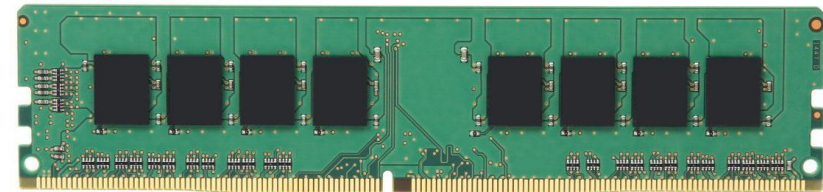
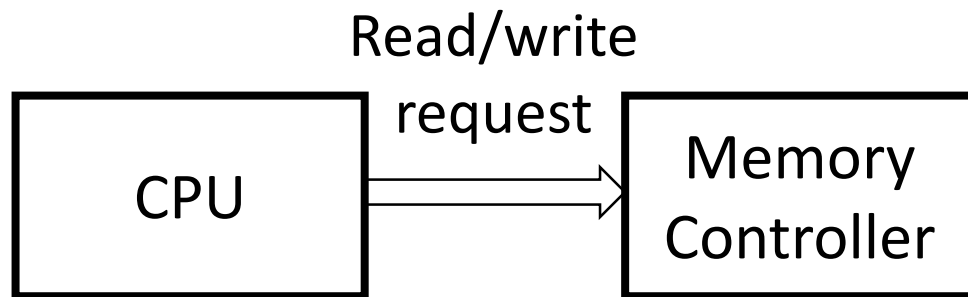
DRAM Functionality: Initialization



4. Memory Controller sets DIMM's Mode Registers

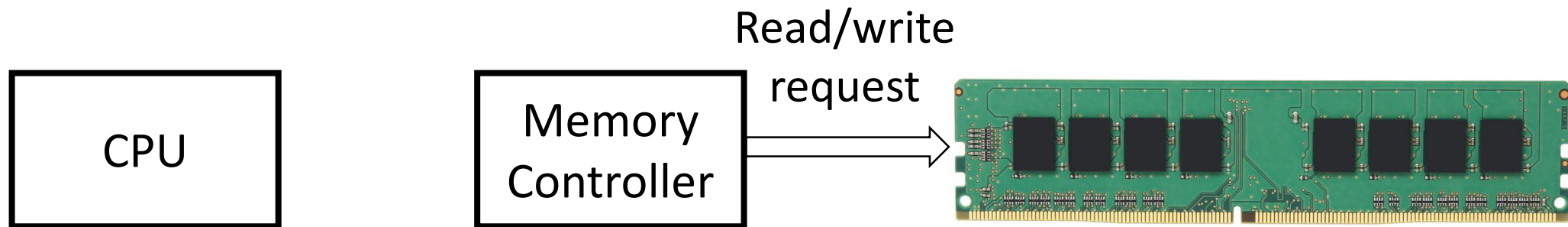
- Enable/disable features, fine tune timings

DRAM Functionality: Normal Operation (S0)



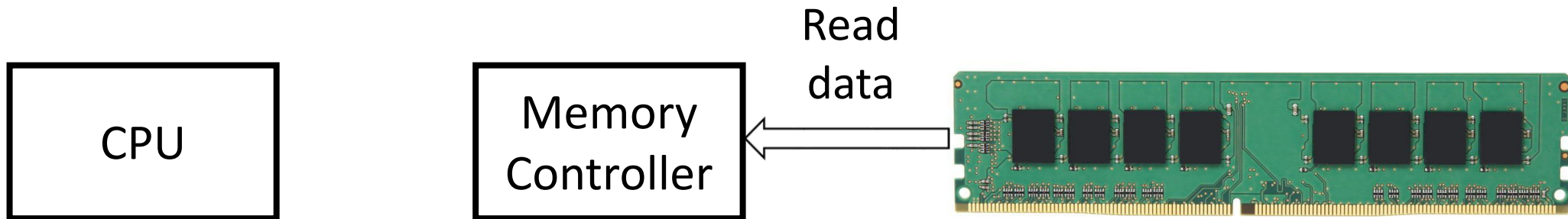
- CPU requests reads & writes
- Memory Controller schedules and optimizes memory accesses
- Memory Controller schedules periodic memory refreshes

DRAM Functionality: Normal Operation (S0)



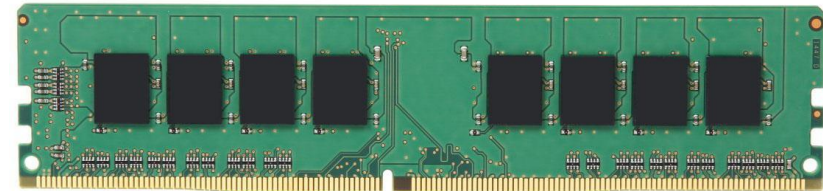
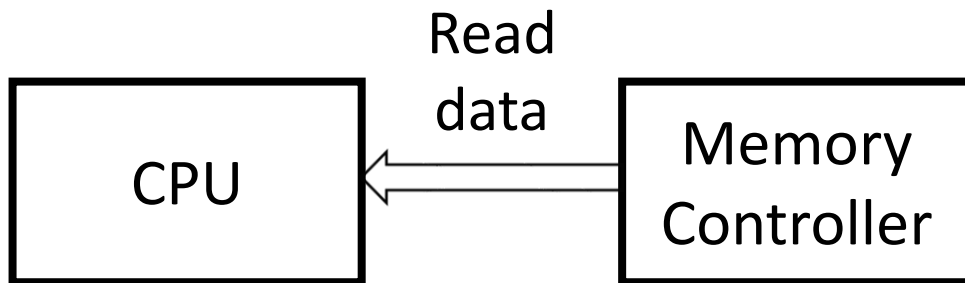
- CPU requests reads & writes
- Memory Controller schedules and optimizes memory accesses
- Memory Controller schedules periodic memory refreshes

DRAM Functionality: Normal Operation (S0)



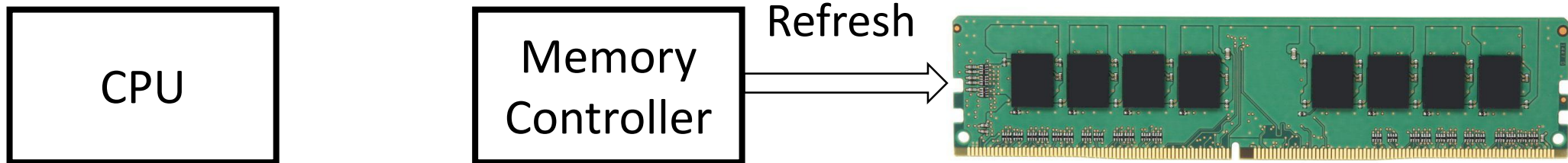
- CPU requests reads & writes
- Memory Controller schedules and optimizes memory accesses
- Memory Controller schedules periodic memory refreshes

DRAM Functionality: Normal Operation (S0)



- CPU requests reads & writes
- Memory Controller schedules and optimizes memory accesses
- Memory Controller schedules periodic memory refreshes

DRAM Functionality: Normal Operation (S0)



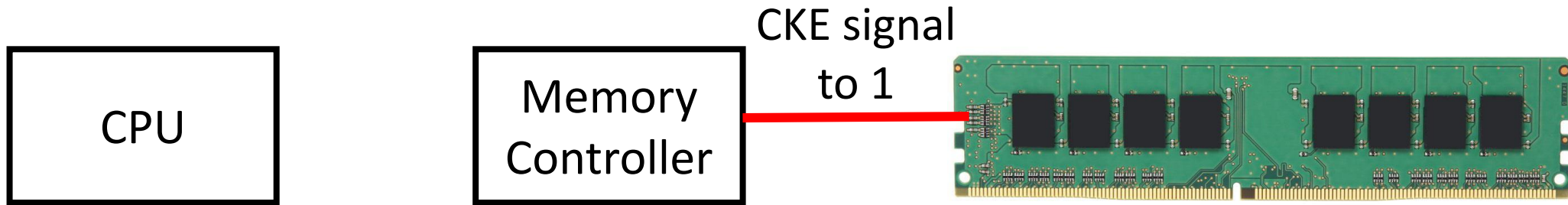
- CPU requests reads & writes
- Memory Controller schedules and optimizes memory accesses
- Memory Controller schedules periodic memory refreshes

DRAM Functionality: Sleep (S3)



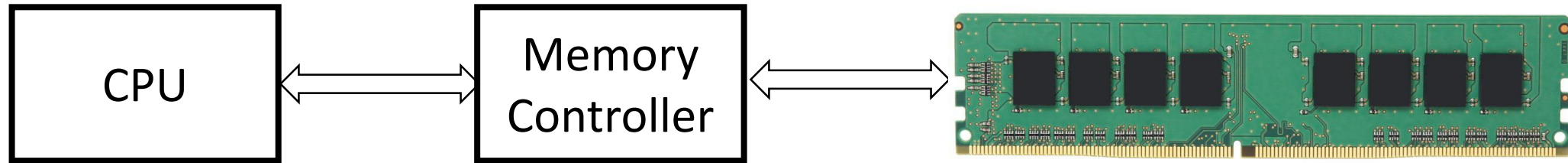
- CPU is powered off
- Memory controller is powered off
 - Clock Enable (CKE) memory signal is 0
 - The rest of the signals are in tri-state
- DIMM is in self-refresh state

DRAM Functionality: Waking up to S0



- CPU and Memory Controller are powered on
- Memory Controller pulls CKE to 1
- Read & Write requests can now be issued

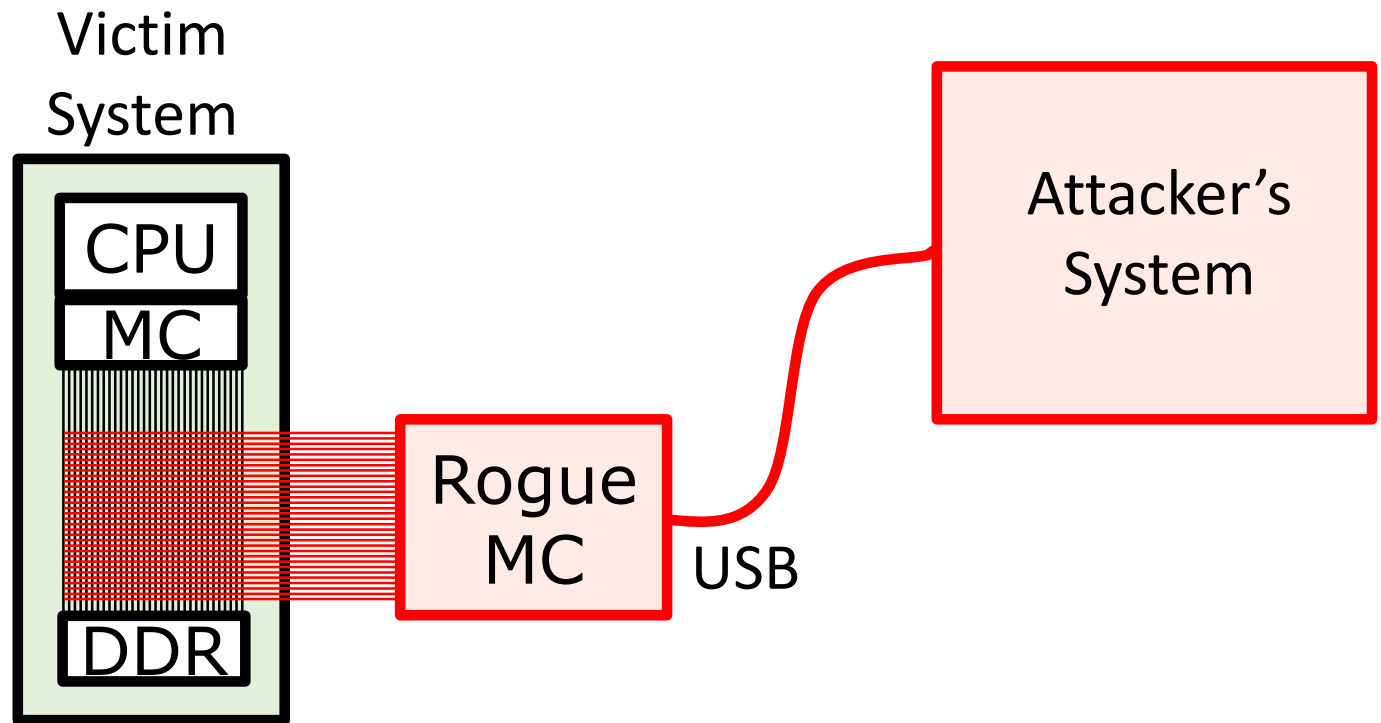
Protocol Limitation



- No state information is held on the DIMM
 - e.g. calibration results, clock frequency
- No authentication between MC and DIMM
- **Nothing prevents an attacker from impersonating a Memory Controller**

Could we exploit that?

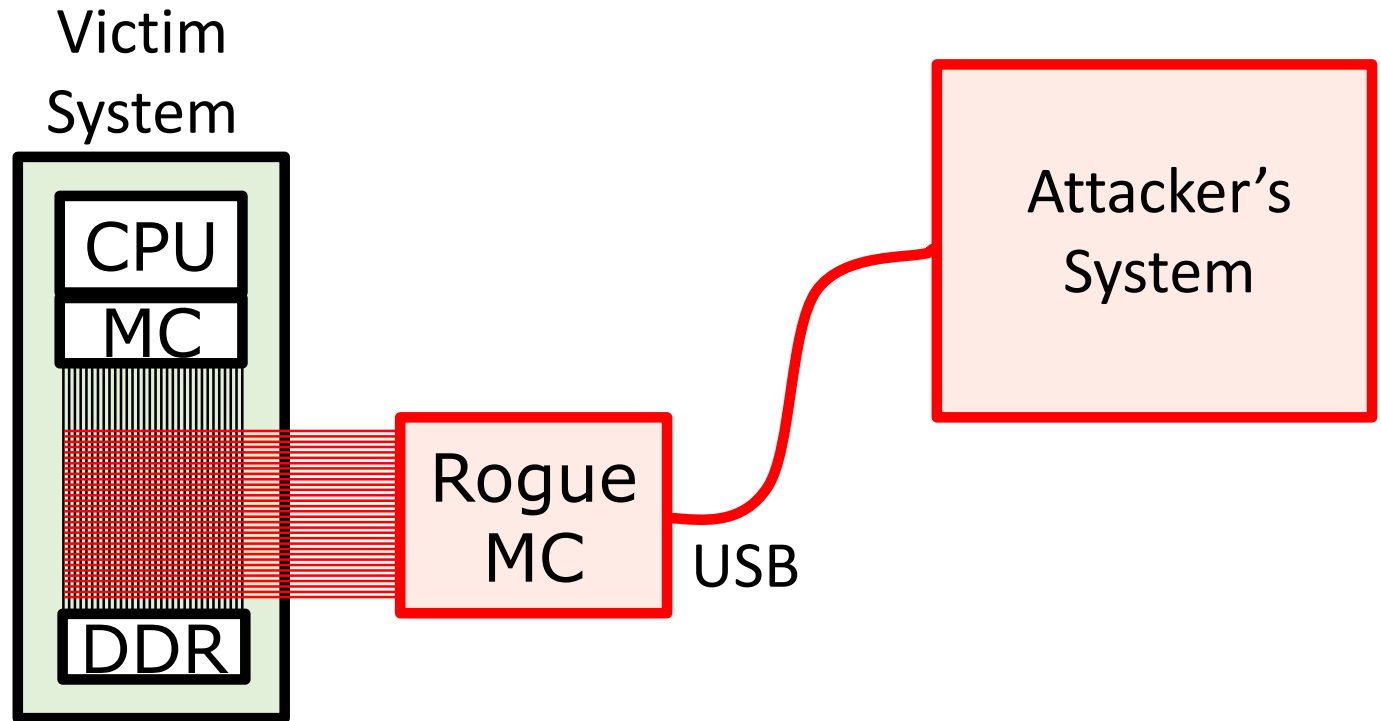
- Create a device that:
 - Attaches to the exposed signal pins
 - Impersonate a MC
- Extremely difficult when system is in S0
- Possible when system is in S3 sleep



Could we exploit that?

- Example:

- The victim goes to a conference and leaves their system in sleep
- The attacker attaches their rogue MC on the exposed pins
- The attacker now owns the system

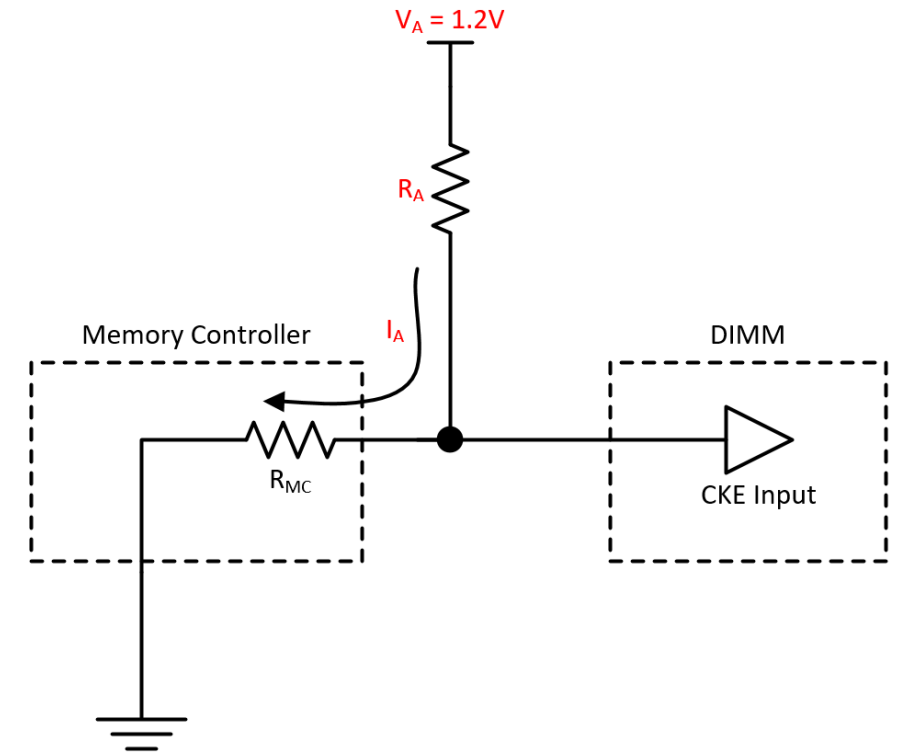


Requirements for Successful Exploitation

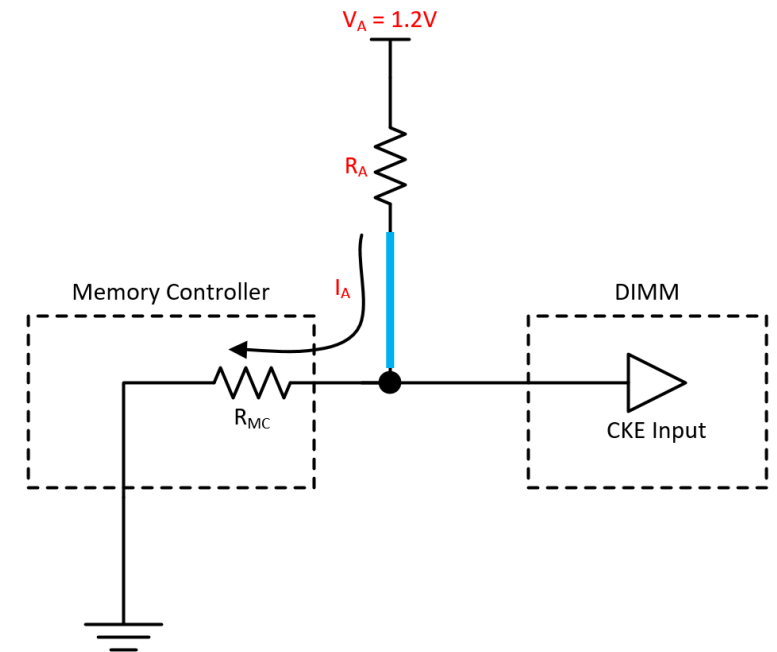
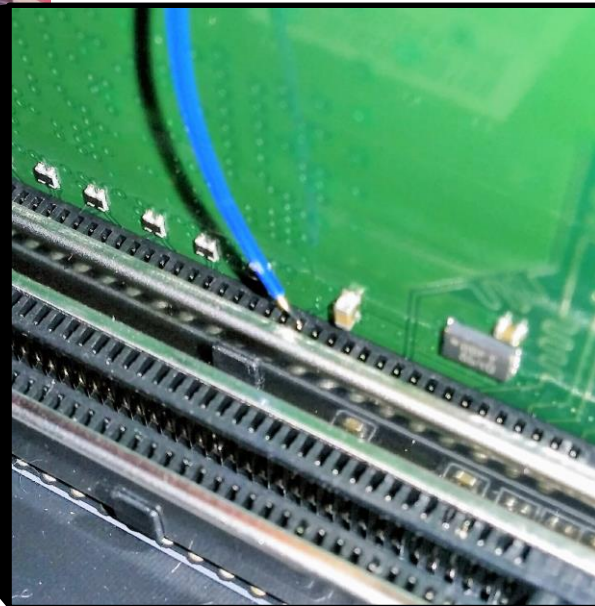
- Attach rogue MC to the victim system while in S3 sleep
 - Pull CKE to 1 to “wake up” the DIMM
 - Calibrate our rogue MC to the memory bus traces
 - Send memory read/write commands to the DIMM
 - Put CKE to 0 to put the DIMM to sleep
- Detach from the victim system and wake up normally

Signal Injection: CKE signal

- CKE must be pulled up above $\sim 0.7V$ (logic high threshold)
- Find R_A so that DIMM sees logic high transition
- Larger values reduce current into victim MC (I_A)



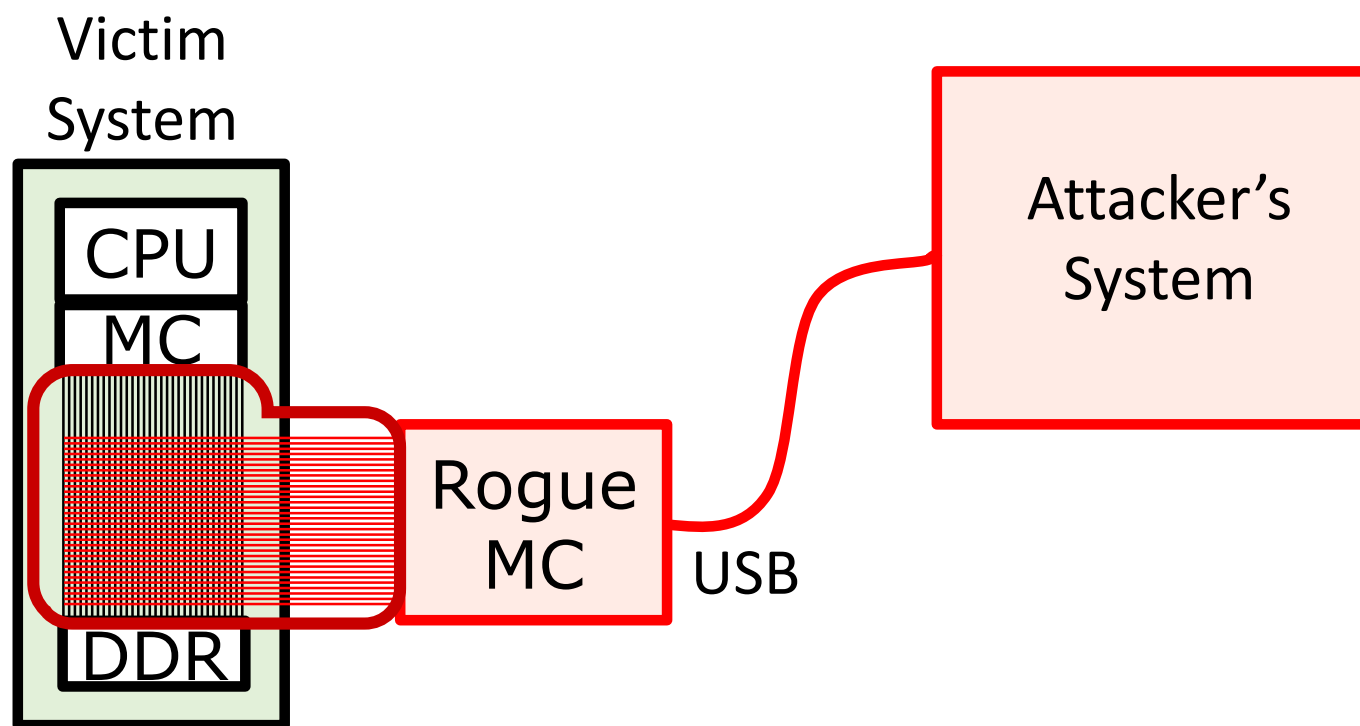
Signal Injection: CKE signal



A safe value for R_A was found to be $\sim 50\text{ohm}$

Rogue MC Calibration

- Need to train our rogue MC with the new traces
- Normally happens on first boot, can be done any time
- Calibration results will be the same on the same system + DIMM models

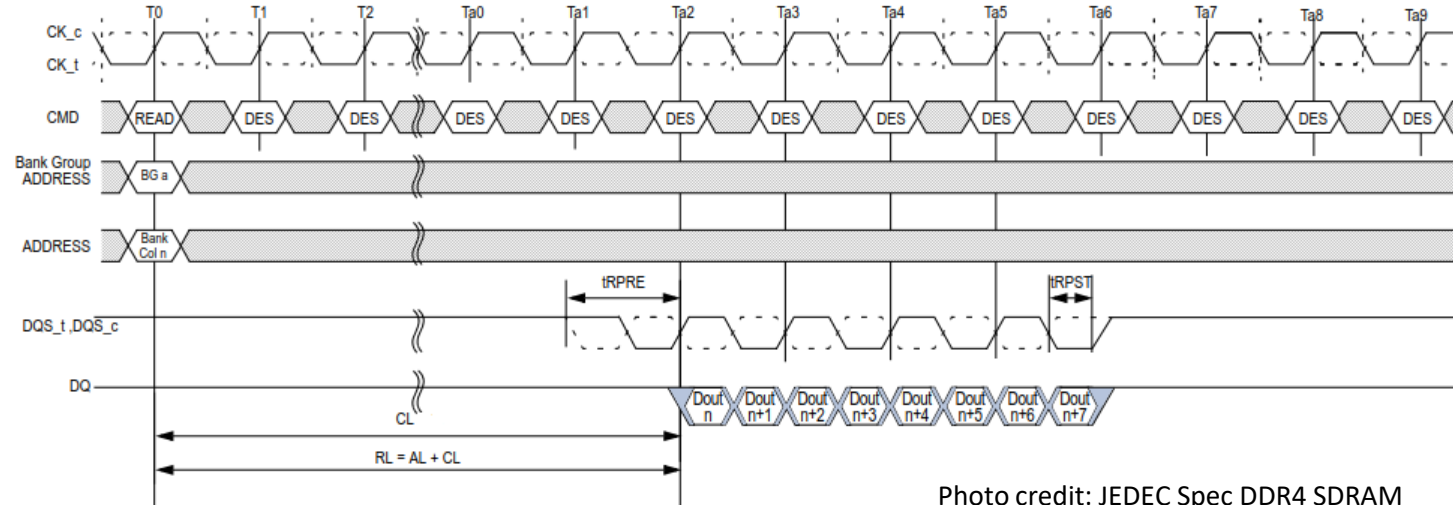


Adjust Clock Frequency

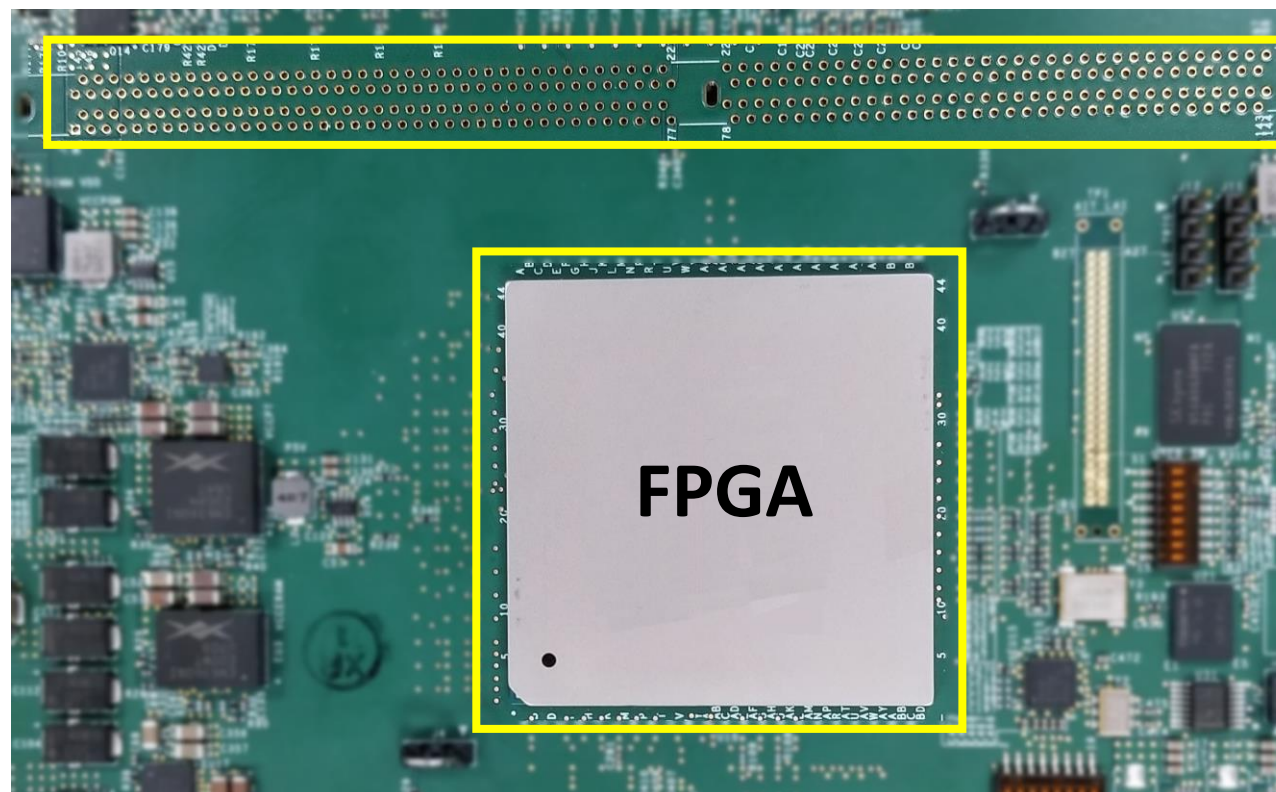
- Rogue MC provides a differential clock signal
- Attacker's clock can be different than the victim clock
 - Lower clock frequency → Better signal integrity
- DDR4 has a default minimum frequency of 800 MHz
 - When "DLL-off" Mode Register is enabled, clock frequency can be significantly lower (as low as ~128 MHz)
 - This Mode Register is accessible to the attacker!

Issue Reads/Writes

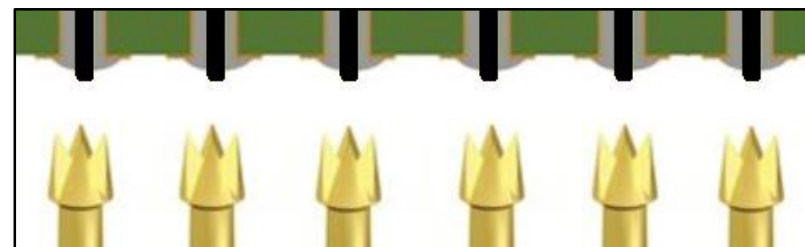
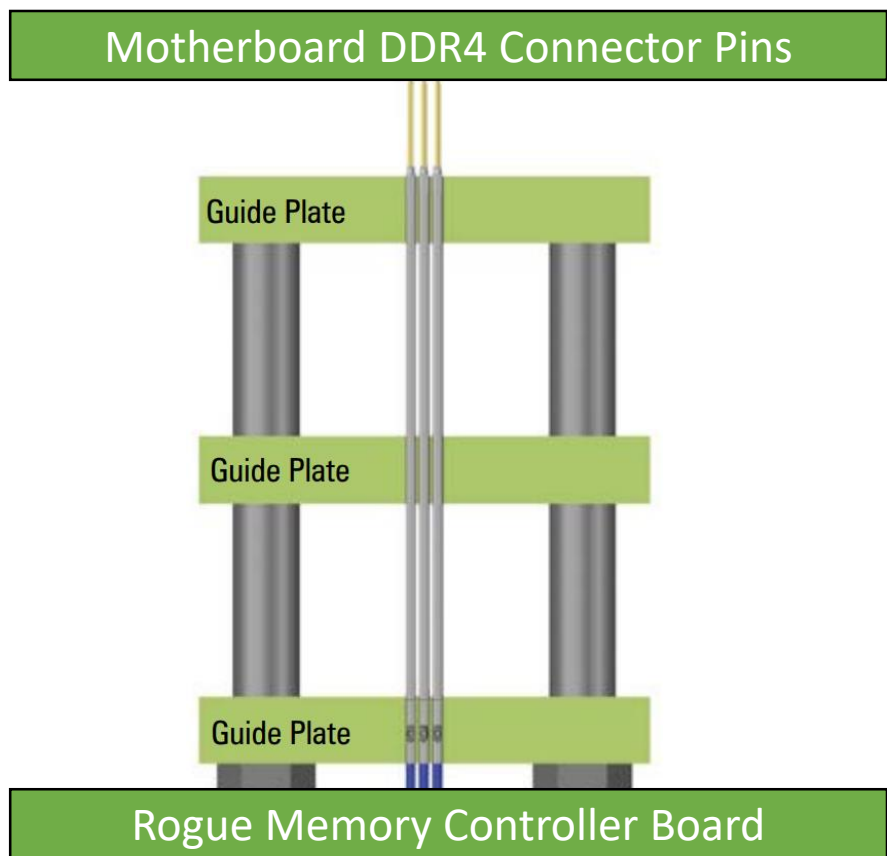
- Follows JEDEC specification using off the shelf FPGA MC
- Iterate through reading all memory addresses
- Selectively write to memory addresses of interest



Rogue Memory Controller Board



Mechanical Connector



Crown Tipped Spring Probes

Threat Model

- Requires a skilled HW attacker
- Physical access to the victim system
 - Invasive
 - Requires time to conduct
- May require prior knowledge of the victim system spec

Challenges / Future Work

- Complete mechanical connector
- Run attack against full length bus
- We anticipate signal noise/reflection challenges

Mitigations

- Short-term
 - Use hibernate (S4) instead of sleep (S3)
 - Focus on physical security
- Long-term
 - Enhance DIMM socket and motherboard design to restrict access to pins
 - Enhance JEDEC spec to perform authentication

Conclusions

- Described early research into a physical attack on DRAM memory
- A skilled attacker could exploit HW design and protocol limitations
- New line of research for physical security
- Short- and long-term mitigations are possible

Questions?

- Thank you!
- We are hiring!

Acknowledgements

We would like to thank the following current or former Intel Employees for their contributions to the work upon which this research builds:

Abhishek Basak

Rodrigo Branco

Sergej Deutsch

David Durham

Ken Grewal

Shay Gueron

Przemek Guzy

Jayson Strayer