從晶片資安漏洞事件談硬體安全議題

中原大學電子工程學系
黃世旭教授

中原大學
Chung Yuan Christian University

Outline

- 硬體安全的重要性
- 電路設計與硬體安全的關係
- Meltdown 及 Spectre 之硬體安全問題
- 電路設計考慮硬體安全的困難
- 結語

Outline

- 硬體安全的重要性
- 電路設計與硬體安全的關係
- Meltdown 及 Spectre 之硬體安全問題
- 電路設計考慮硬體安全的困難
- 結語

Hardware Security 範疇

- Hardware
  - Integrated Circuits (ICs)
  - Field Programmable Gate Arrays (FPGAs)
  - Embedded Systems
- Security
  - Vulnerabilities, Threats, Attacks from Hardware
  - Hardware for Security
  - The Security in Hardware Design Process

Hardware Security 重要性

- 最近有關 CPU 之 out-of-order execution 及 speculative execution 設計漏洞衍生的 Meltdown 與 Spectre 兩大攻擊手法，衝擊全球數十億電腦(桌機、筆電)、伺服器甚至手機的處理器，恐導致記憶體資料外洩。此新聞事件，正突顯硬體安全的重要性。
- 隨著物聯網應用持續蓬勃發展，可以預期市場對於硬體安全的需求日益增長。

Hardware Security 範疇

- Hardware
  - Integrated Circuits (ICs)
  - Field Programmable Gate Arrays (FPGAs)
  - Embedded Systems
- Security
  - Vulnerabilities, Threats, Attacks from Hardware
  - Hardware for Security
  - The Security in Hardware Design Process

Naive Attacker Model

- Attacks on the channel between communicating parties
  - lousy cryptography
  - insecure protocols
- End-points are regarded as black boxes
- Protection: mathematically strong algorithms and protocols

參考文獻：E. Poll, “Attacker Models for Hardware (aka Embedded Security)” https://www.cs.ru.nl/E.Poll/hw/slides/2_smartcards_intro.pdf

Improved Attacker Model

- Attacks on the channel or on end-points
  - sloppy users
  - software bugs
- Protection
  - mathematically strong algorithms and protocols
  - secure software implementations

參考文獻：E. Poll, “Attacker Models for Hardware (aka Embedded Security)” https://www.cs.ru.nl/E.Poll/hw/slides/2_smartcards_intro.pdf

Hardware Security Attacker

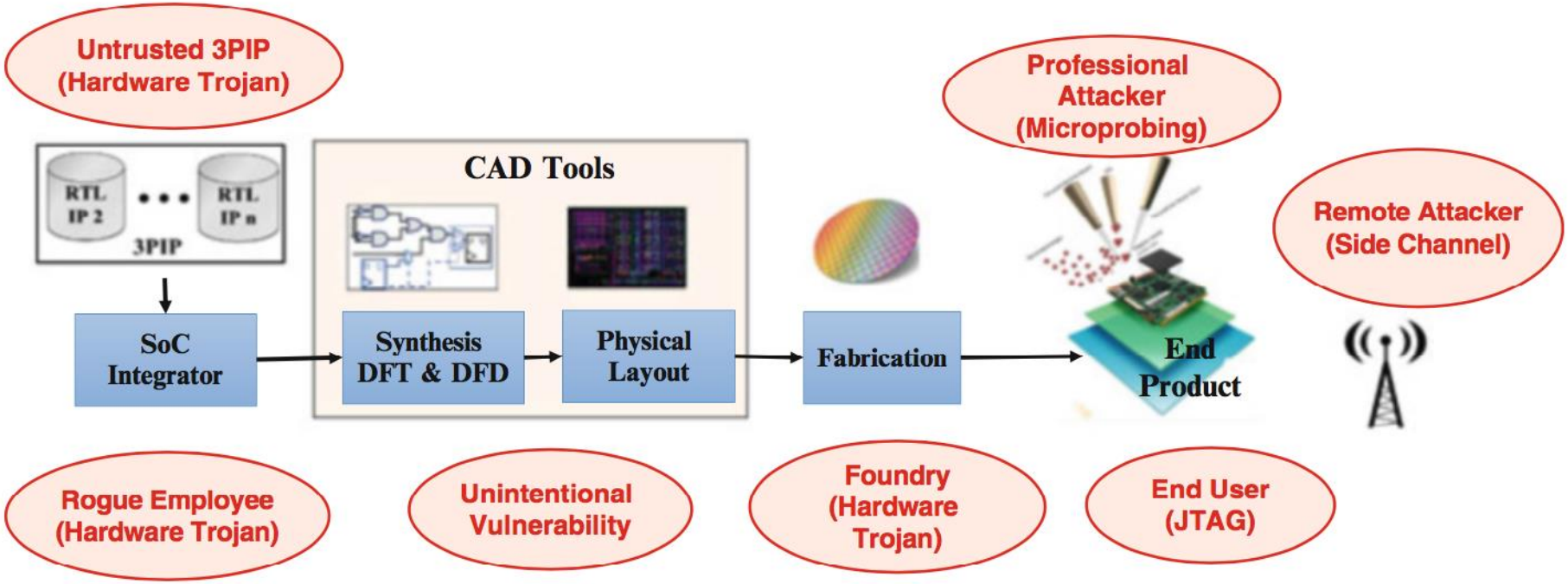
- Attacks on the channel or on end-points
  - sloppy users
  - software bugs
  - Hardware weakness
- Protection
  - mathematically strong algorithms and protocols
  - secure software implementations
  - secure hardware implementations

參考文獻：E. Poll, “Attacker Models for Hardware (aka Embedded Security)” https://www.cs.ru.nl/E.Poll/hw/slides/2_smartcards_intro.pdf

Outline

- 硬體安全的重要性
- 電路設計與硬體安全的關係
- Meltdown 及 Spectre 之硬體安全問題
- 電路設計考慮硬體安全的困難
- 結語

Potential Adversaries in SoC Design Process



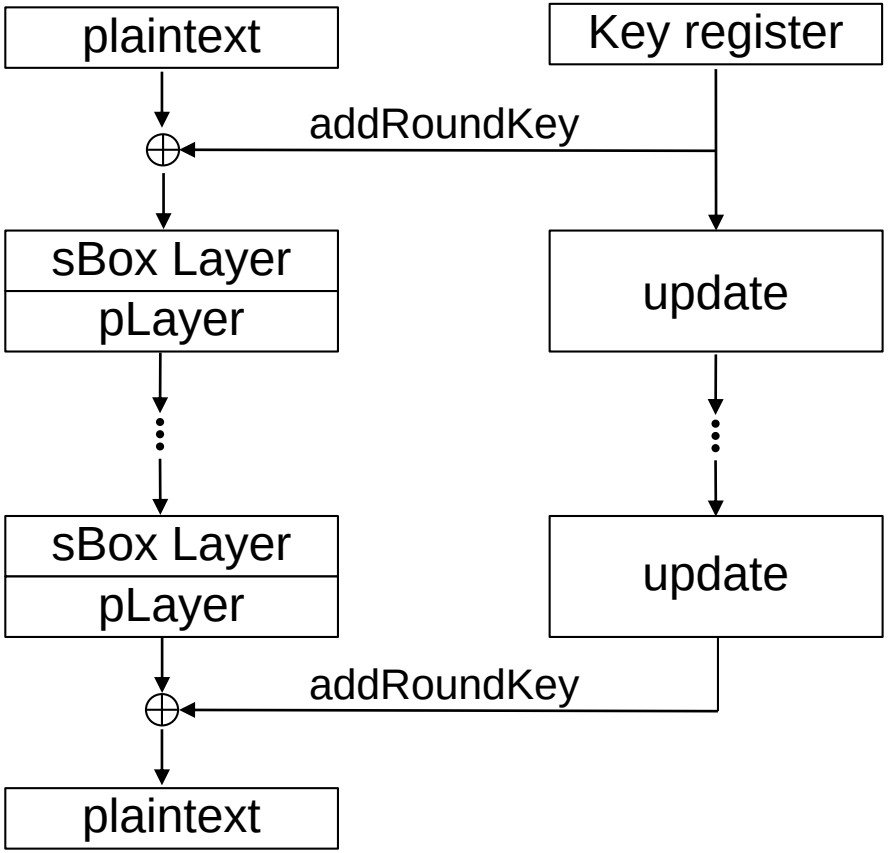
圖片來源：P. Mishra, et al. (editors), “Hardware IP Security and Trust”, Springer International Publishing, 2017.

Sources of Vulnerabilities

- Design Mistakes
  - a vulnerability may be unintentionally created in the hardware implementation
- CAD Tools
  - don't-care conditions can create vulnerabilities in the circuit
- Design for Testability (DFT)
  - the increased controllability and observability can create numerous vulnerabilities

參考文獻：P. Mishra, et al. (editors), "Hardware IP Security and Trust", Springer International Publishing, 2017.

Design Mistake




```
module PRESENT_ENCRYPT (
                output [63:0] odat,   //  data output
                input  [63:0] idat,   //  data input
                input  [79:0] key,    //   key input
                input         load,   //   data load
                input         clk     //    clock
    };

//------------wires,   registers-----------------------
reg     [79:0]   kreg;                  //   key register
reg     [63:0]   dreg;                  //   data register



//   Load/reload key into key register
always @(posedge clk)
begin
   if   (load)
        kreg   <=   key;
   else
        kreg   <=   kdat2;
end
```

參考文獻：P. Mishra, et al. (editors), “Hardware IP Security and Trust”, Springer International Publishing, 2017.

Vulnerabilities in Combinational Logics

A	B	C	X	Y
0	0	0	--	--
0	0	1	1	0
0	1	0	0	1
0	1	1	--	--
1	0	0	1	1
1	0	1	--	--
1	1	0	--	--
1	1	1	--	--

Vulnerabilities in Combinational Logics

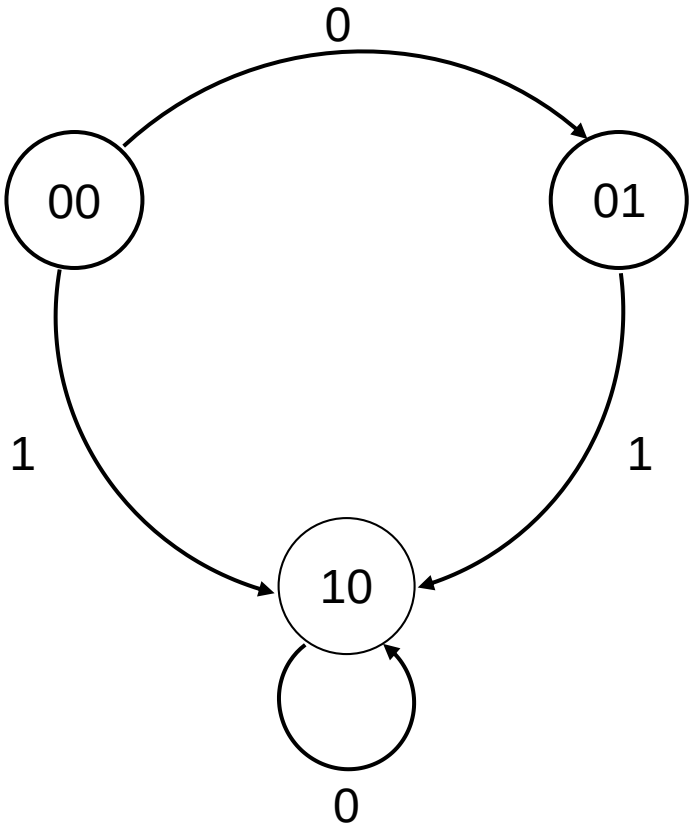
A	B	C	X	Y
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

$X = B'$ $Y = C'$

Backdoor
ABC = 101

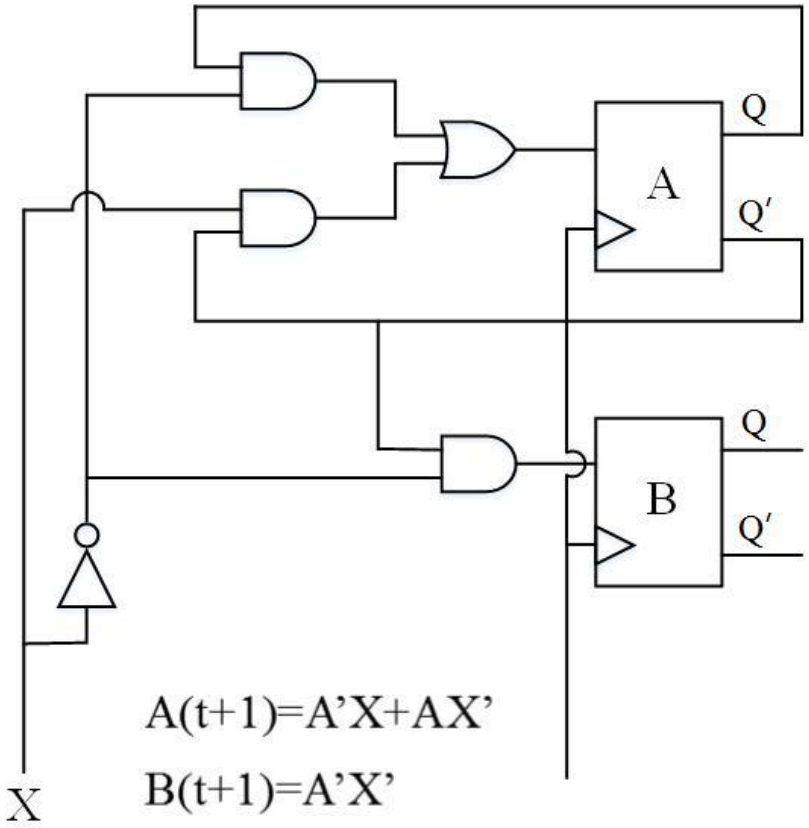
Fault Injection
ABC = 011 or 111

Vulnerabilities in Sequential Logics



Present State		Input	Next State	
A(t)	B(t)	X	A(t+1)	B(t+1)
0	0	0	0	1
0	0	1	1	0
0	1	1	1	0
1	0	0	1	0

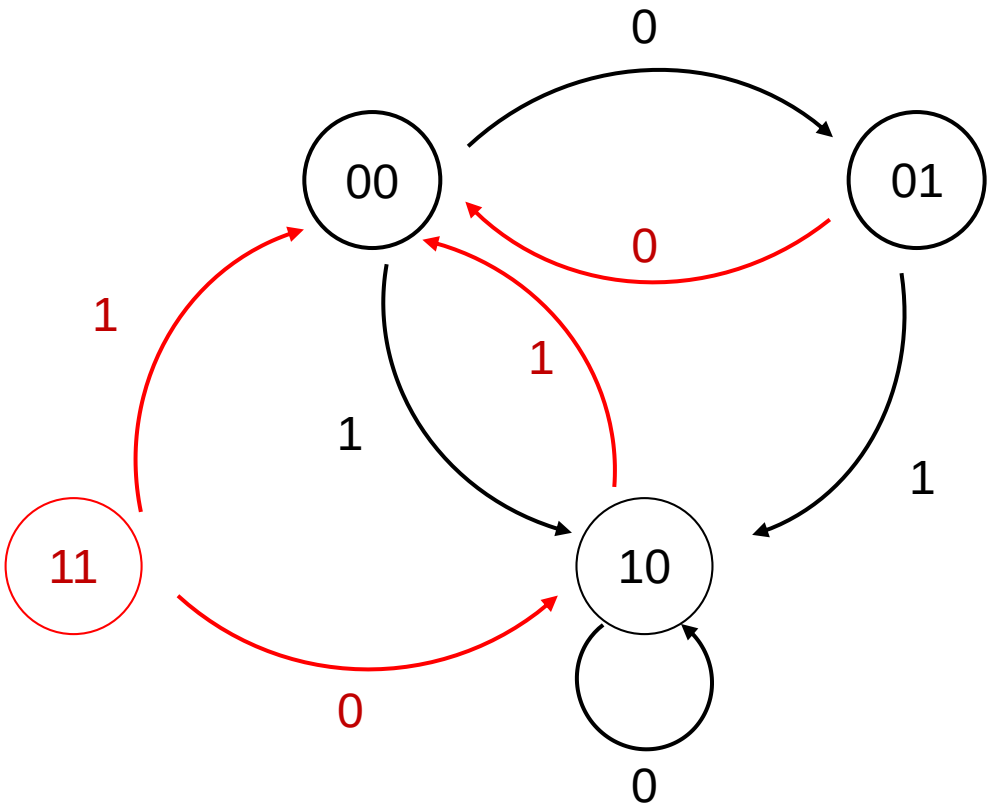
Vulnerabilities in Sequential Logics



Vulnerabilities in Sequential Logics

Present State		Input	Next State	
A(t)	B(t)	X	A(t+1)	B(t+1)
0	0	0	0	1
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	1	0
1	0	1	0	0
1	1	0	1	0
1	1	1	0	0

Vulnerabilities in Sequential Logics



Outline

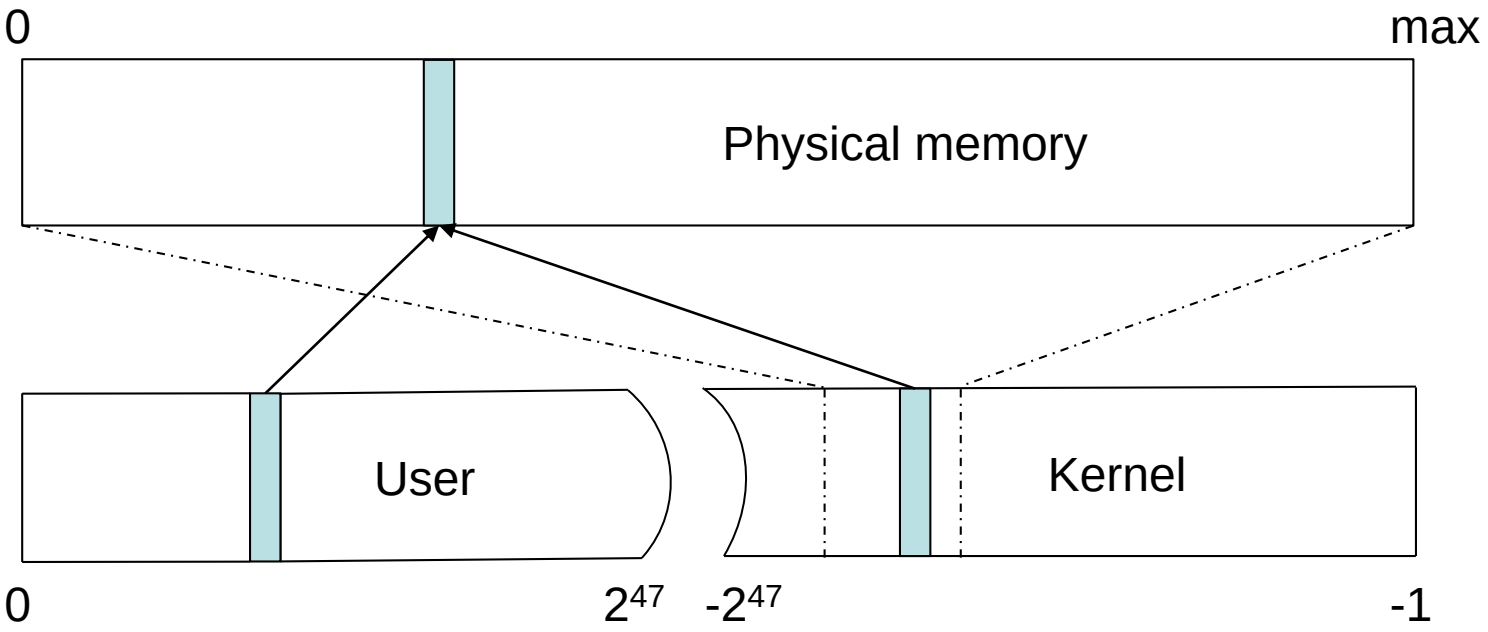
- 硬體安全的重要性
- 電路設計與硬體安全的關係
- Meltdown 及 Spectre 之硬體安全問題
- 電路設計考慮硬體安全的困難
- 結語

Memory Isolation

- In modern processors, the isolation between the kernel and user processes is typically realized by a supervisor bit of the processor that defines whether a memory page of the kernel can be accessed or not.
- In practice, there is no change of the memory mapping when switching from a user process to the kernel.

參考文獻：M. Lipp, et al., "Meltdown". https://arxiv.org/abs/1801.01207

Memory Page Sharing



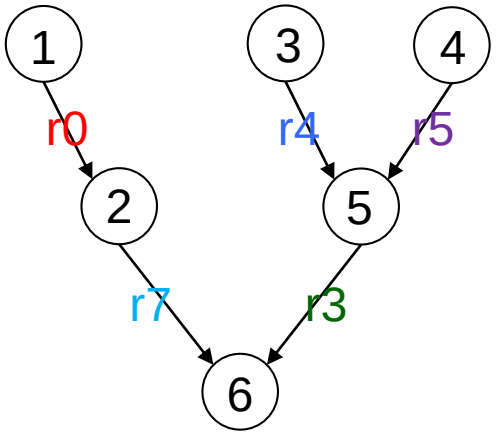
圖片來源：M. Lipp, et al., "Meltdown". https://arxiv.org/abs/1801.01207

Out-of-order Execution

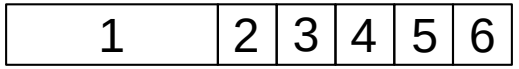
- Out-of-order execution is an important feature of modern processors in order to overcome latencies of busy execution units.
  - a memory fetch unit needs to wait for data arrival from memory
- Instead of stalling the execution, modern processors run operations out-of-order.
  - look ahead and schedule subsequent operations to idle execution units

Out-of-order Execution

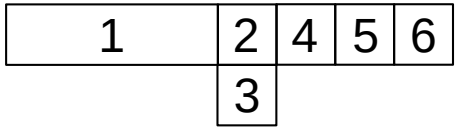
(1)	r0	←	r3	/	r6
(2)	r7	←	r0	+	r1
(3)	r4	←	r4	+	1
(4)	r5	←	r5	−	r2
(5)	r3	←	r4	+	r5
(6)	r6	←	r7	*	r3



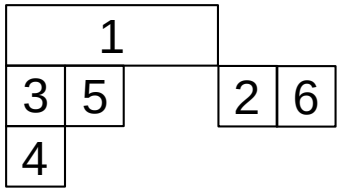
In-order execution



In-order (2-way superscalar)

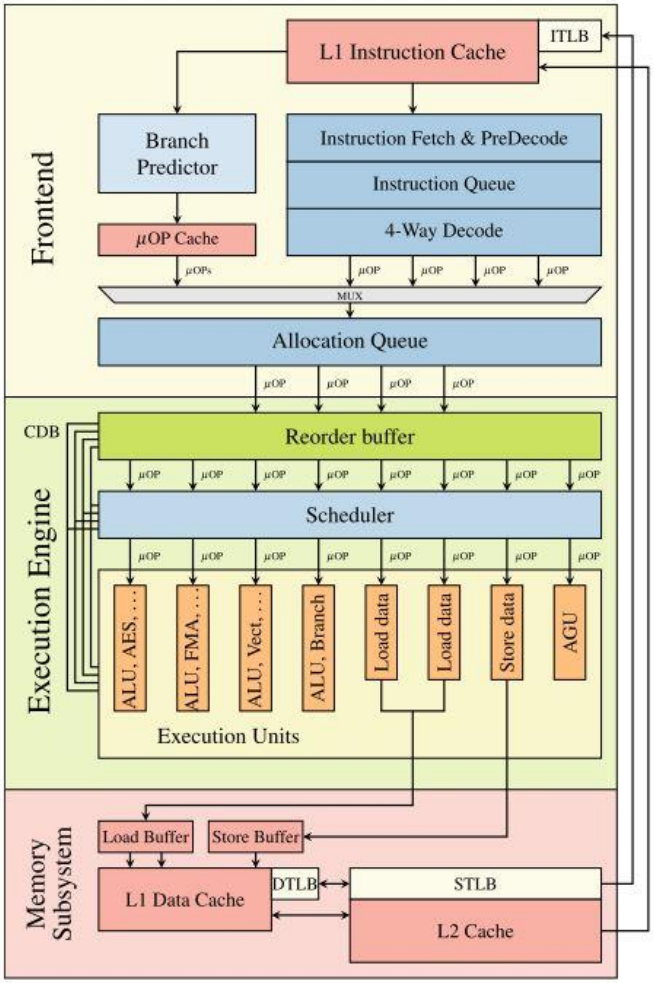


Out-of-order execution



中原大學
Chung Yuan Christian University

Intel's Skylake Microarchitecture



圖片來源：M. Lipp, et al., "Meltdown". https://arxiv.org/abs/1801.01207

Meltdown Attack

- Meltdown are based on the side effects caused by out-of-order execution.
- Meltdown allows an adversary to obtain a dump of the entire kernel address space, including any mapped physical memory.
- Meltdown does not exploit any software vulnerability.

參考文獻：M. Lipp, et al., "Meltdown". https://arxiv.org/abs/1801.01207

Meltdown Attack

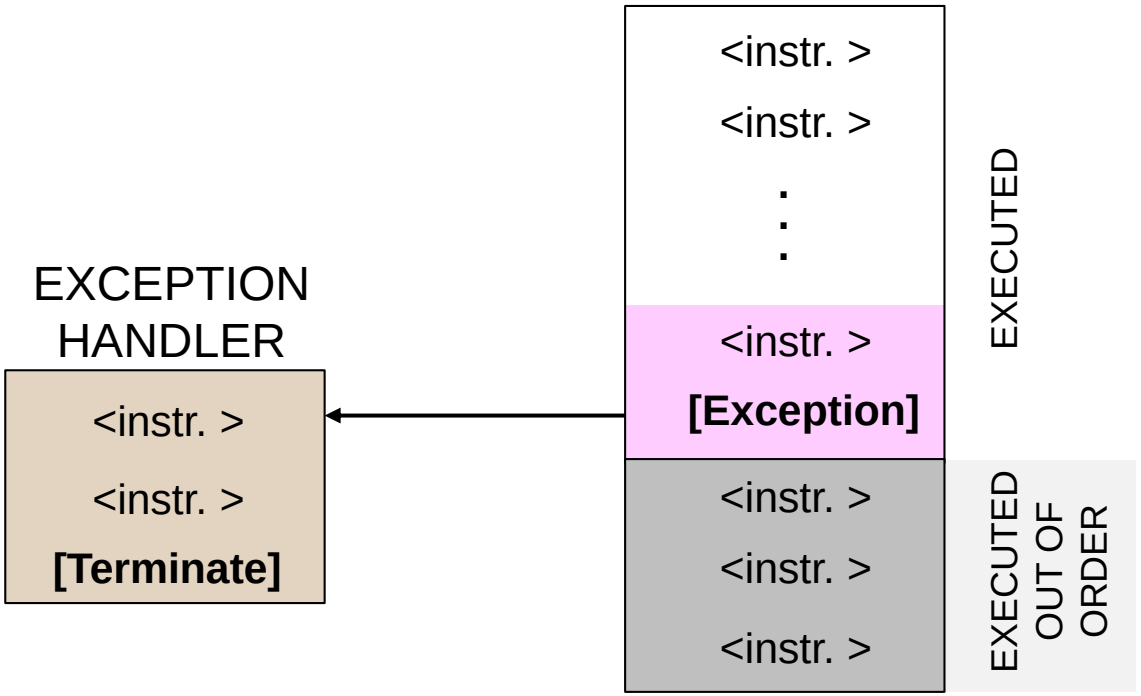
- Out-of-order memory lookups influence the cache.
  - can be detected through the cache side channel.
- An attacker can dump the entire kernel memory
  - read privileged memory in an out-of-order execution stream
  - transmit the data from this elusive state via a micro-architectural covert channel to the outside world.

A Toy Example

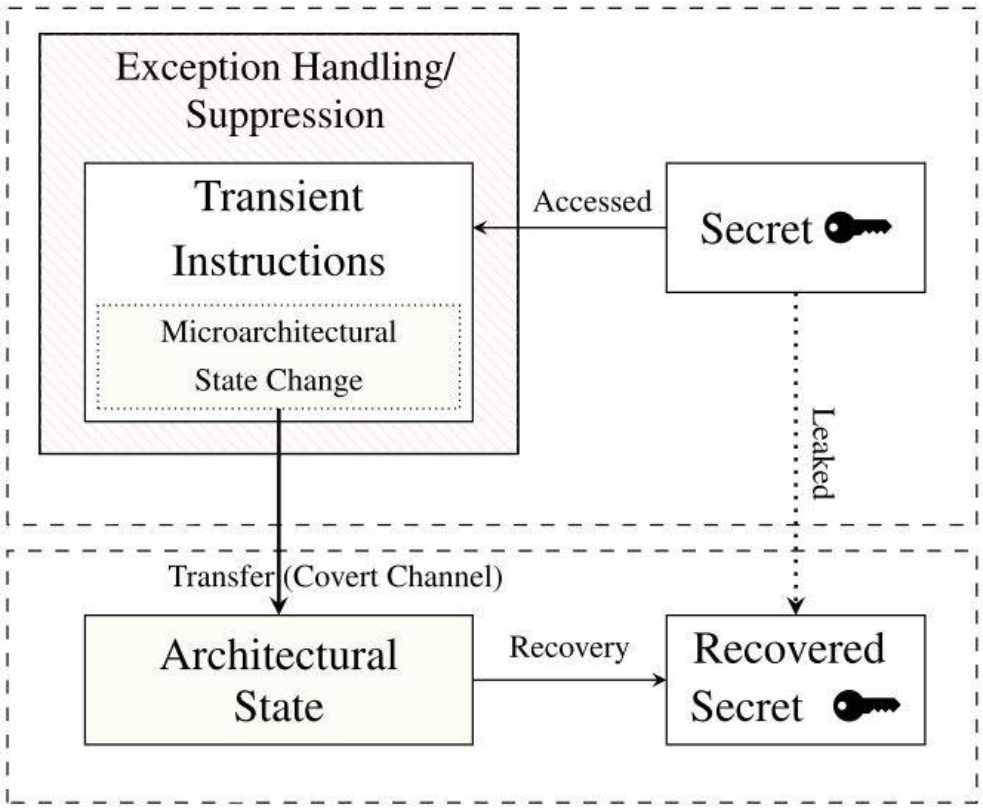
```
1   raise_exception();
2   //  the line below is never reached
3   access(probe_array[data * 4096]);
```

- Due to the out-of-order execution, the CPU might have already executed the following instructions as there is no dependency on the exception.
- During the out-of-order execution, the referenced memory is fetched into a register and is also stored in the cache.

Exception Handling



Meltdown Attack



圖片來源：M. Lipp, et al., "Meltdown". https://arxiv.org/abs/1801.01207

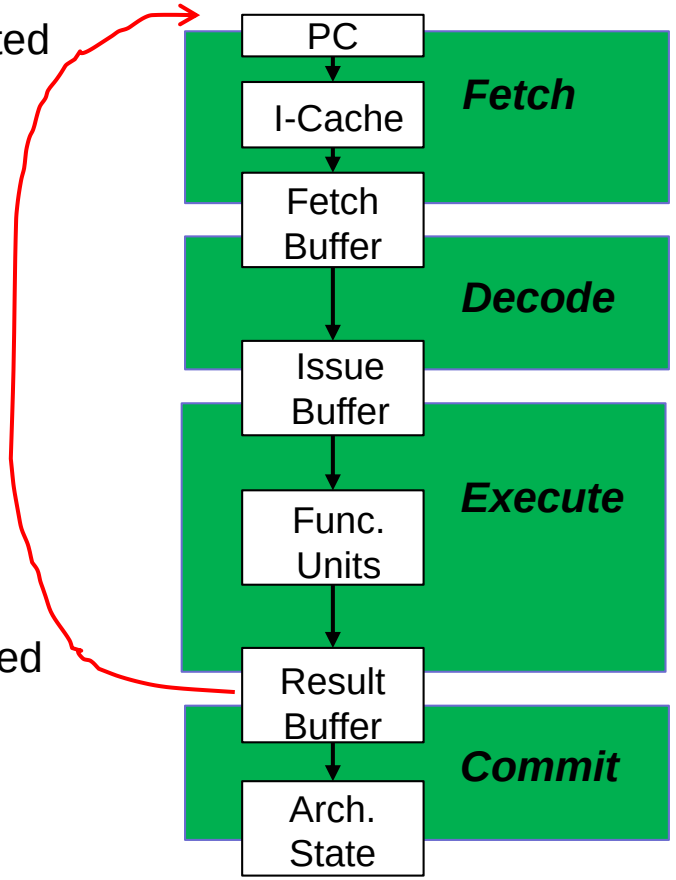
Spectre Attacks

- In modern processors, branch prediction and speculative execution are used to maximize circuit performance.
- Spectre attacks involve:
  - induce a victim to speculatively execute operations that would not be performed during correct program execution
  - leak the victim's confidential information via a side channel to the adversary.

參考文獻：P. Kocher, et al., "Spectre Attacks: Exploiting Speculative Execution". https://arxiv.org/abs/1801.01203

Branch Penalty

Modern processors may have > 10 pipeline stages between next PC calculation and branch resolution



圖片來源：https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-823-computer-system-architecture-fall-2005/lecture-notes/l13_brnchpred.pdf

A Toy Example

```
if  (x < array1_size)
    y = array2[array1[x] * 256];
```

- The value of x is maliciously chosen (and out-of-bounds) such that array1[x] resolves to a secret byte k somewhere in the victim's memory.
- array1_size and array2 are not present in the processor's cache, but k is cached.
- Previous operations received values of x that were valid, leading the branch predictor to assume the if will likely be true.

參考文獻：P. Kocher, et al., "Spectre Attacks: Exploiting Speculative Execution". https://arxiv.org/abs/1801.01203

Instruction Timing

```
if  (false but mispredicts as true)
    multiply R1, R2
multiply R3, R4
```

- **Note that Spectre vulnerabilities do not necessarily need to involve caches.**
- Instructions whose timing depends on the values of the operands may leak information on the operands.

參考文獻：P. Kocher, et al., "Spectre Attacks: Exploiting Speculative Execution". https://arxiv.org/abs/1801.01203

Leveraging arbitrary observable effects

```
if  (x < array1_size)    {
    y = array2[array1[x] * 256];
    // do something using Y that is
    // observable when speculatively executed
}
```

- The timing of when the observable operation begins will depend on the cache status of array2.

參考文獻：P. Kocher, et al., "Spectre Attacks: Exploiting Speculative Execution". https://arxiv.org/abs/1801.01203

Outline

- 硬體安全的重要性
- 電路設計與硬體安全的關係
- Meltdown 及 Spectre 之硬體安全問題
- 電路設計考慮硬體安全的困難
- 結語

電路設計考慮硬體安全的困難

- 許多電路設計者不瞭解資安的需求
- 考慮硬體安全會增加電路成本
  - Timing
  - Area
  - Power
- 需要 CAD 工具支援
- 硬體安全與電路可測性的本質上矛盾

Outline

- 硬體安全的重要性
- 電路設計與硬體安全的關係
- Meltdown 及 Spectre 之硬體安全問題
- 電路設計考慮硬體安全的困難
- 結語

結語

- 晶片資安漏洞事件 (Meltdown 及 Spectre) 所造成的影響，顯示硬體安全的重要性。
- 隨著物聯網應用持續蓬勃發展，可以預期市場對於硬體安全的需求日益增長。
- 電路設計考慮硬體安全面臨一些挑戰，未來需要電路設計領域專家與資訊安全領域專家共同努力。

Thank You!