

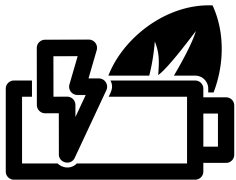
CPU Undervolting

Exploits and Potentials

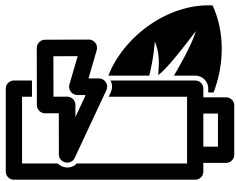
Jonas Juffinger

2024-01-22

CPU Undervolting



Decrease Power Consumption



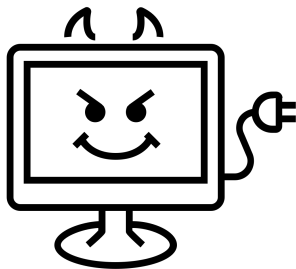
Decrease Power Consumption



Increase Performance



Cause Reliability Issues



Cause **Security** Issues

About Faulting Instructions

An infinite loop...

```
uint64_t multiplier = 0x1276af93a60d1cb7;
uint64_t var = 0x1313f7d45dd0339a * multiplier;

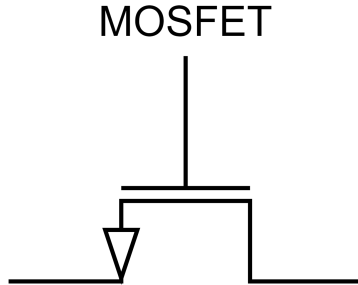
while (var == 0x1313f7d45dd0339a * multiplier)
{
    var = 0x1313f7d45dd0339a;
    var *= multiplier;
}

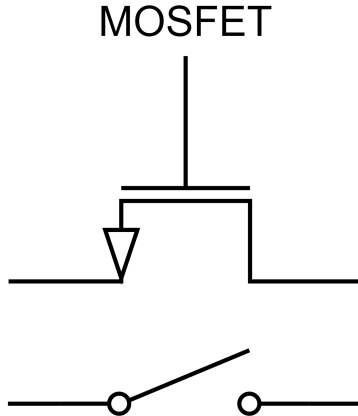
printf("loop exited!\n");
```

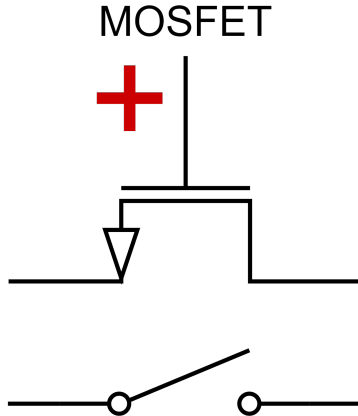

Demo

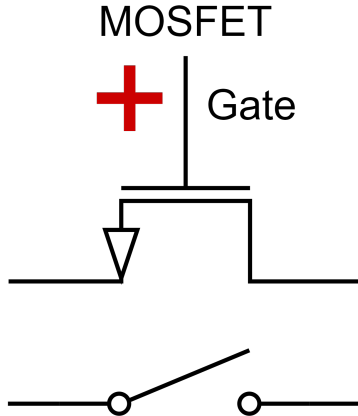
Voltage-Frequency Relationship in CMOS Circuits

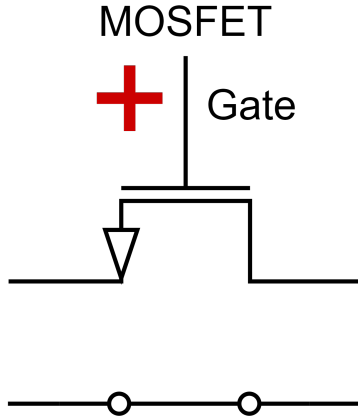


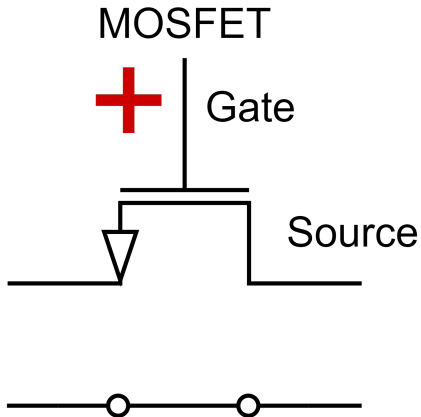


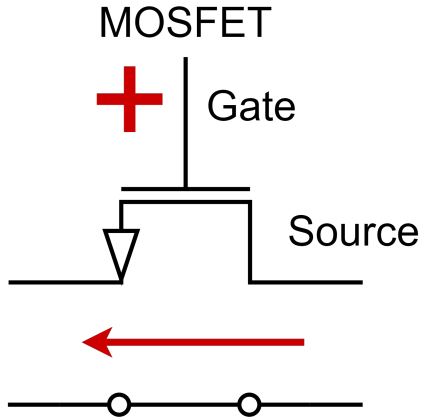


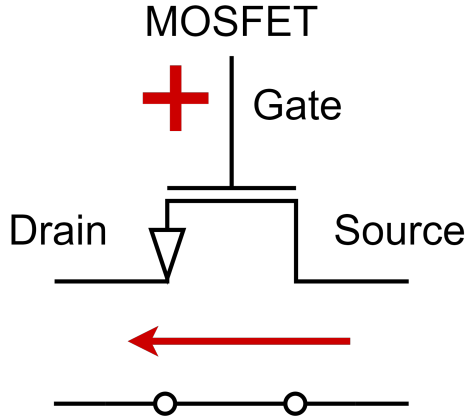


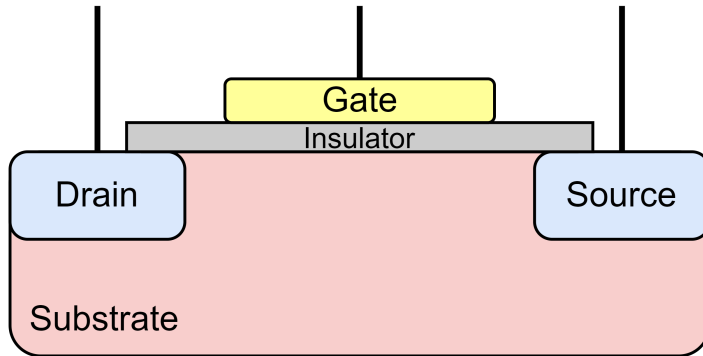


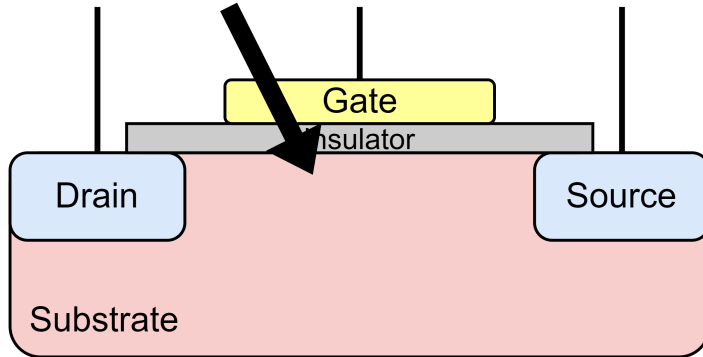


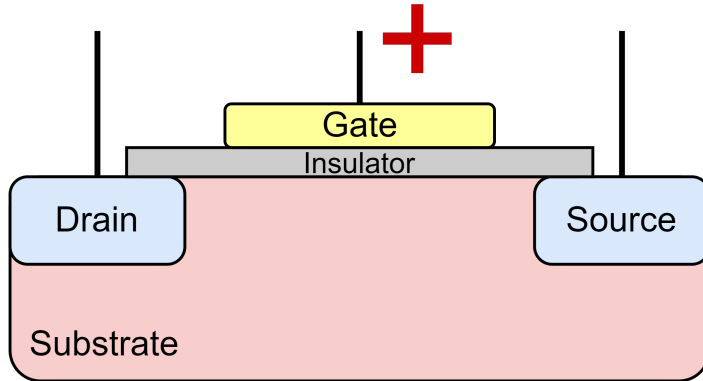


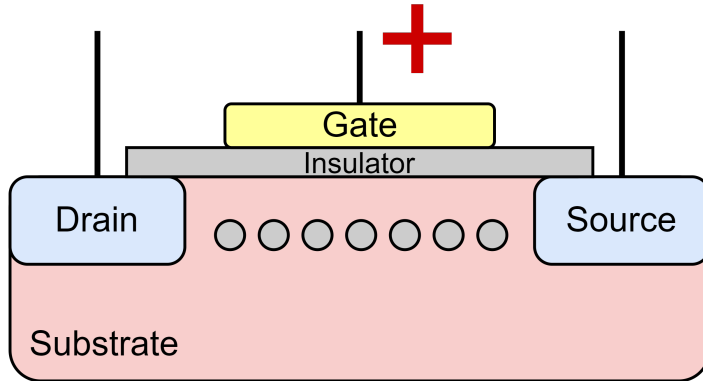


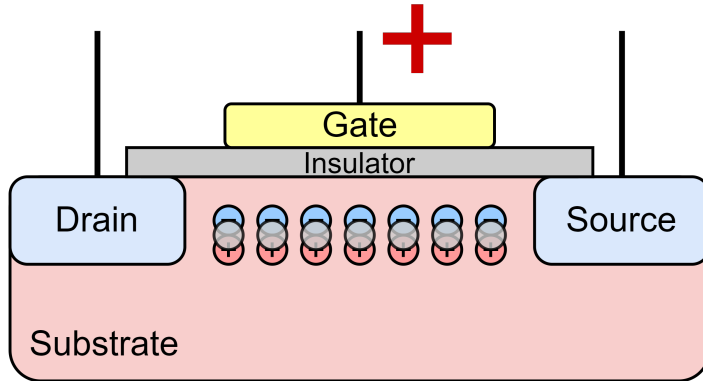


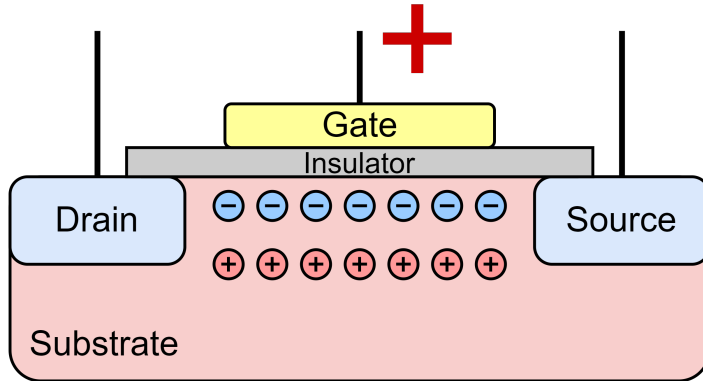


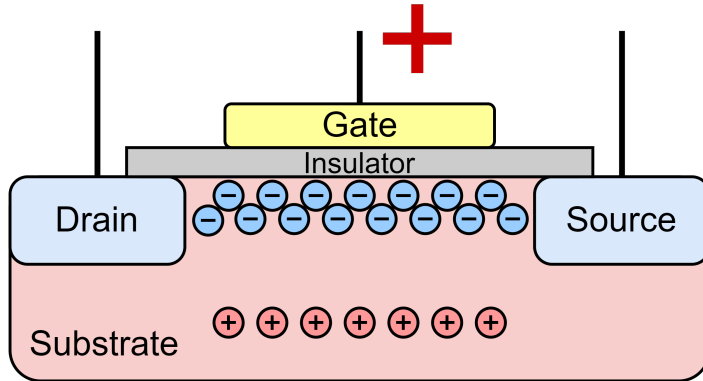


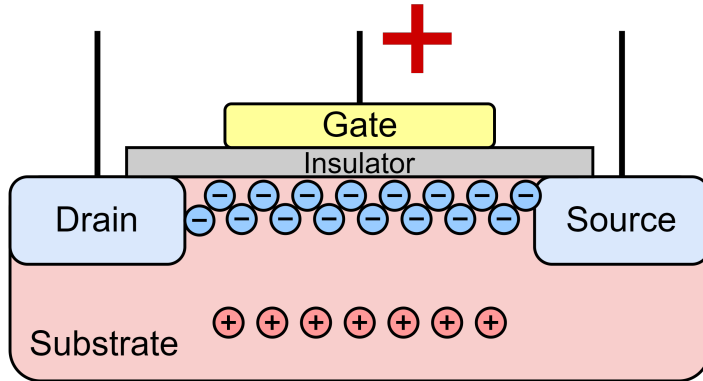


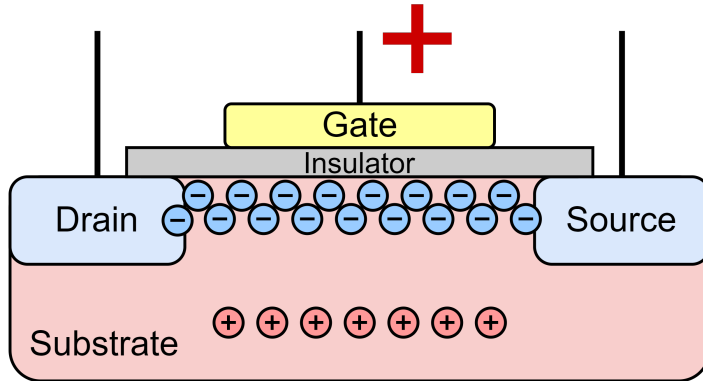


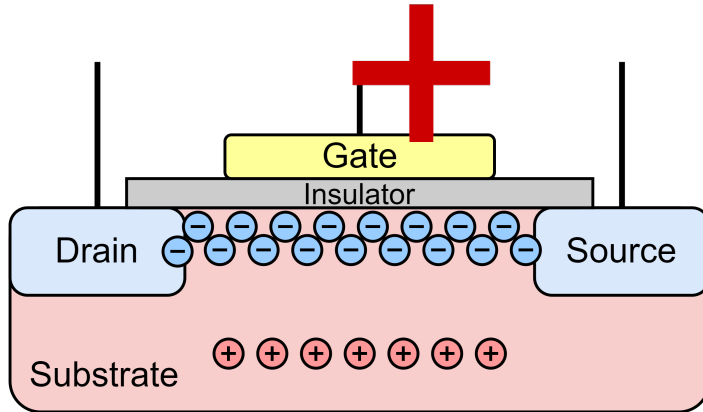


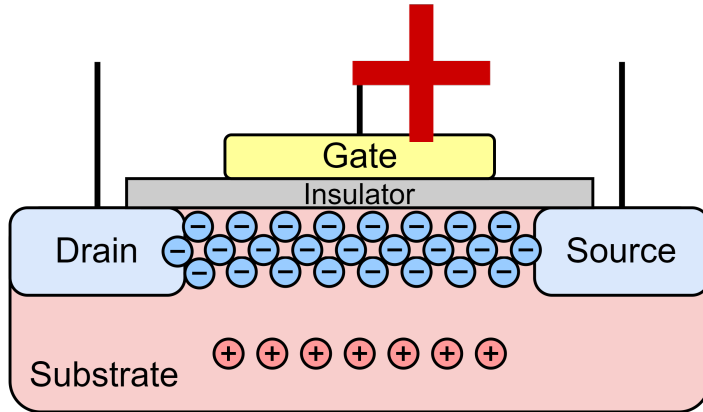


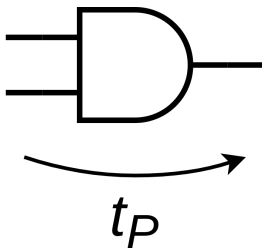


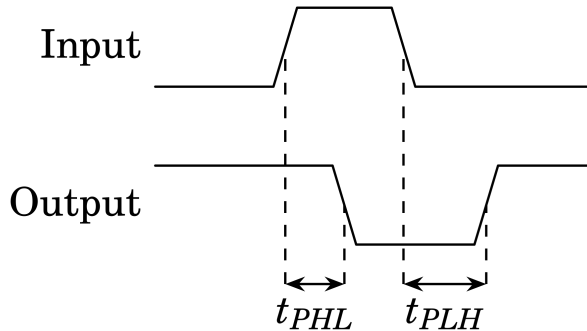




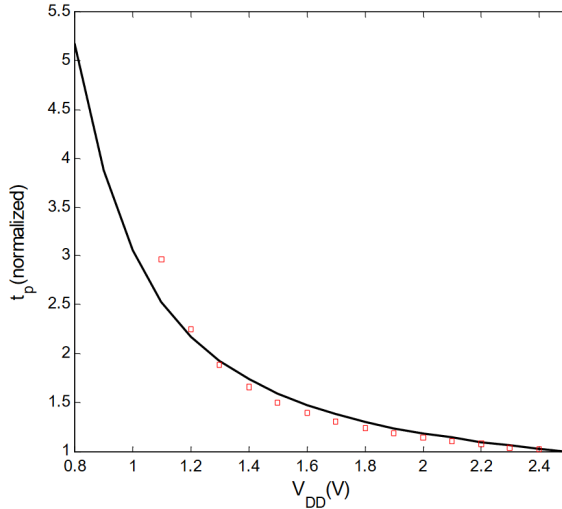




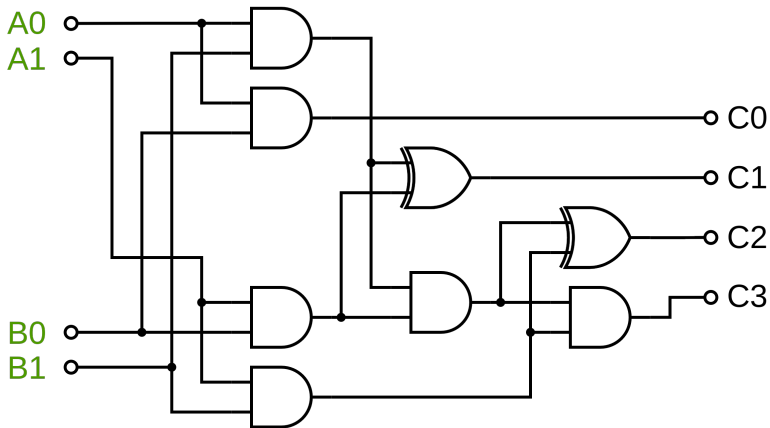




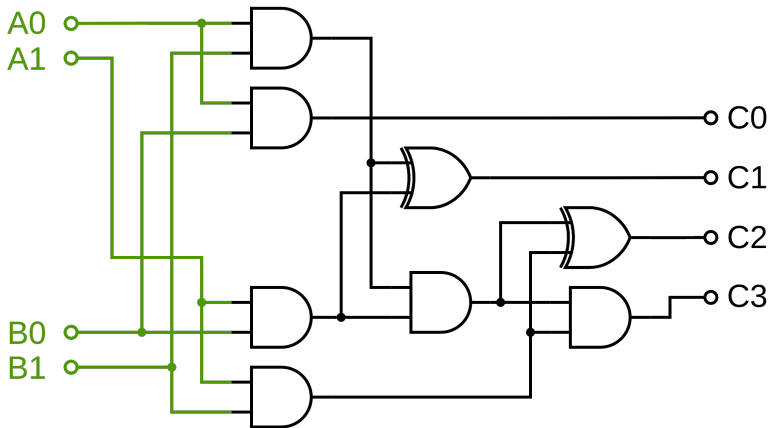
Propagation Delay vs Voltage



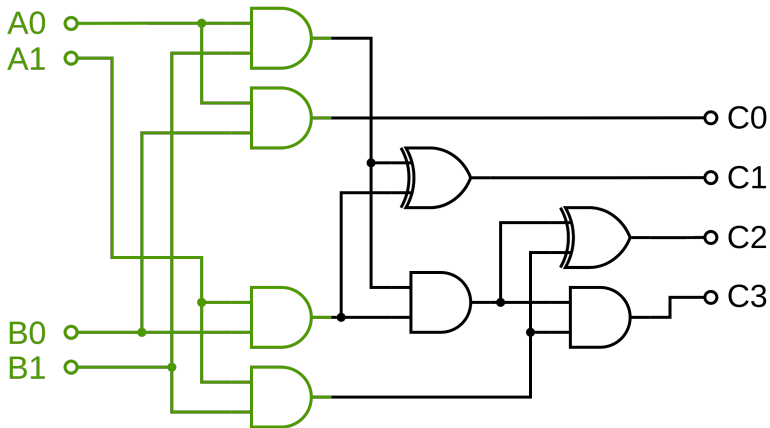
Propagation Delay vs Clock



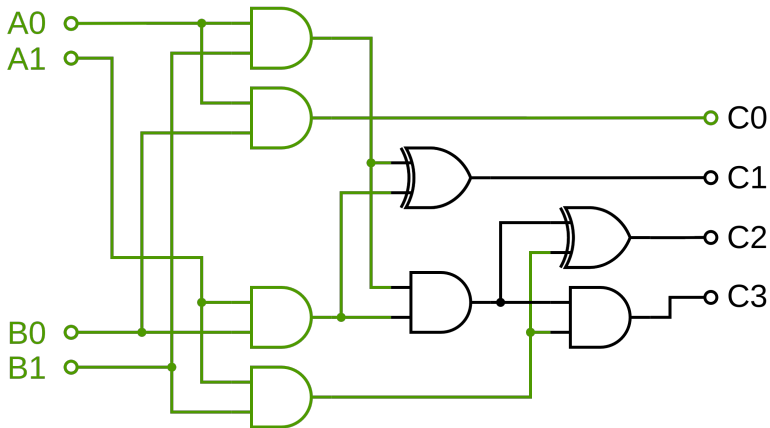
Propagation Delay vs Clock



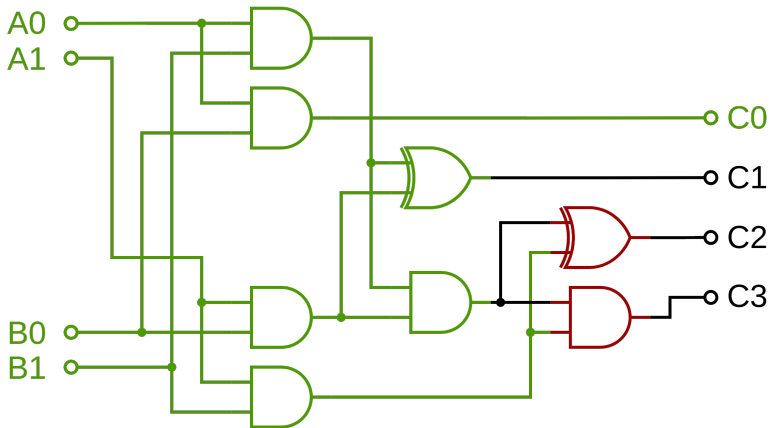
Propagation Delay vs Clock



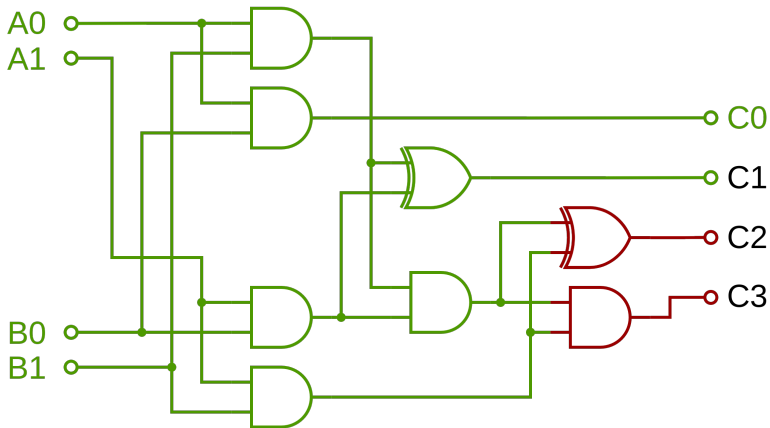
Propagation Delay vs Clock



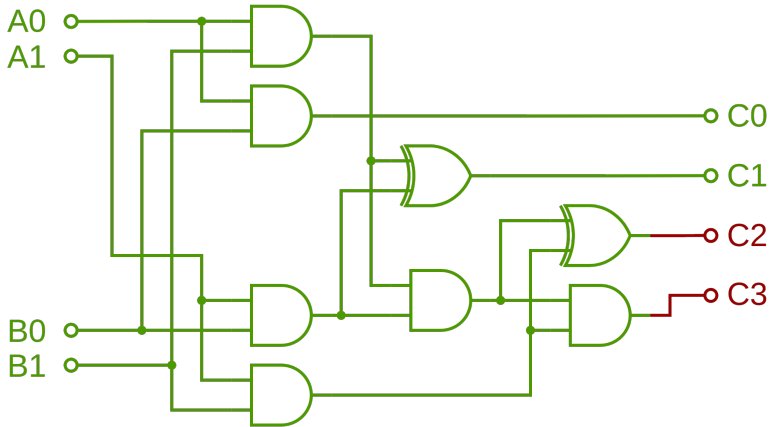
Propagation Delay vs Clock



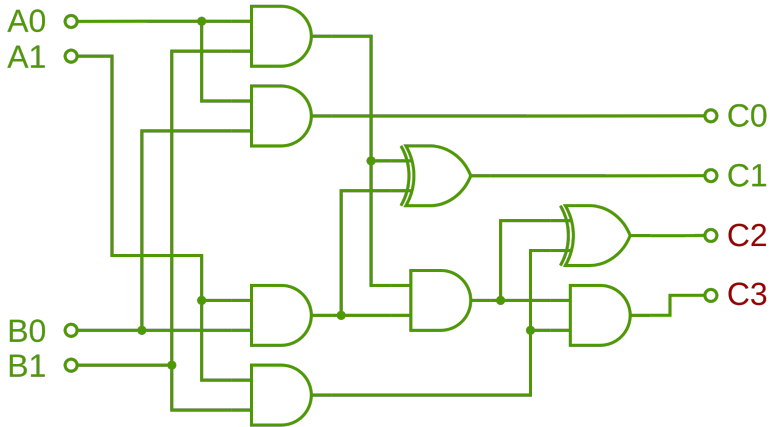
Propagation Delay vs Clock



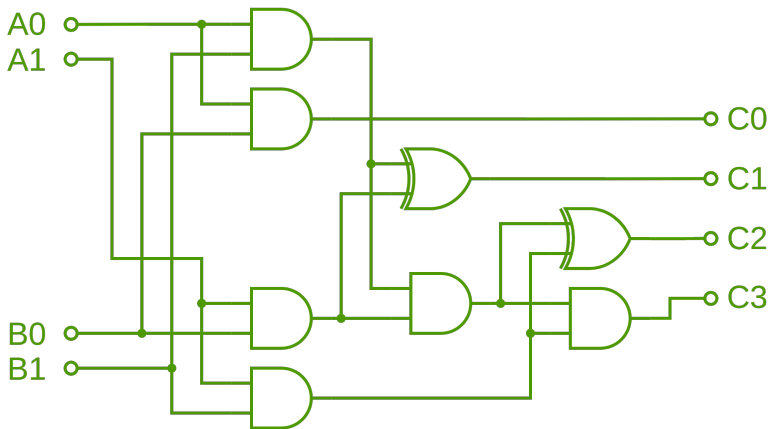
Propagation Delay vs Clock



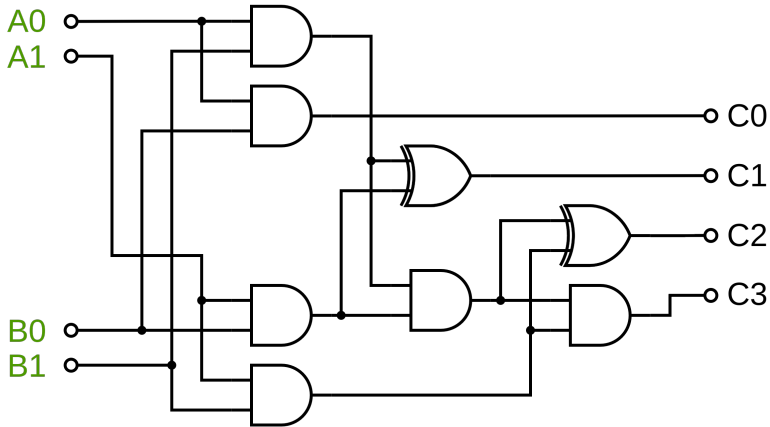
Propagation Delay vs Clock



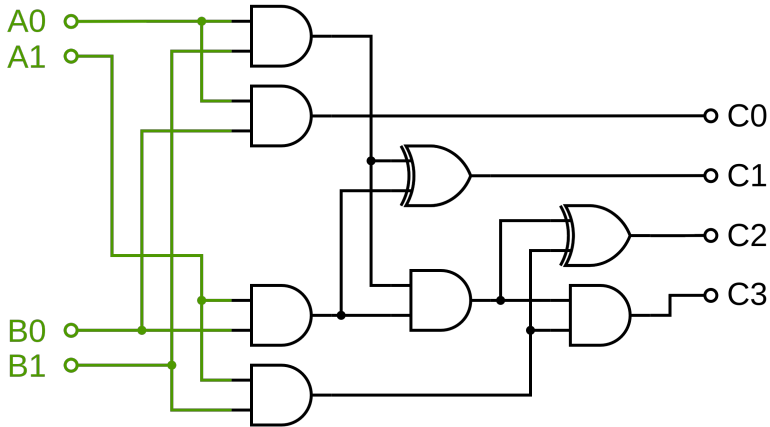
Propagation Delay vs Clock



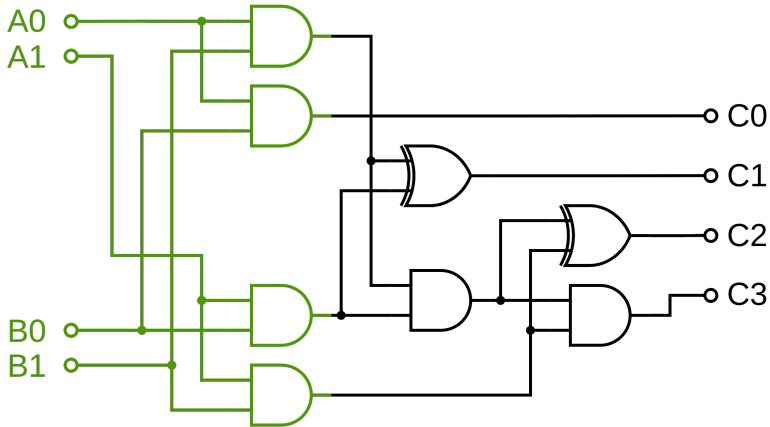
Propagation Delay vs Clock (but now the voltage is too low)



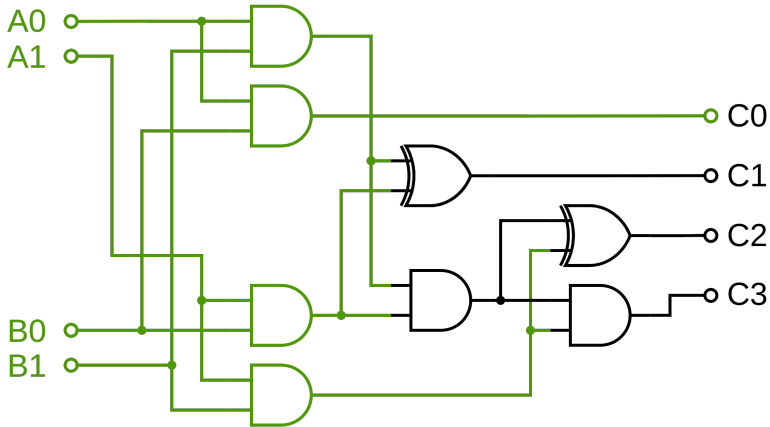
Propagation Delay vs Clock (but now the voltage is too low)



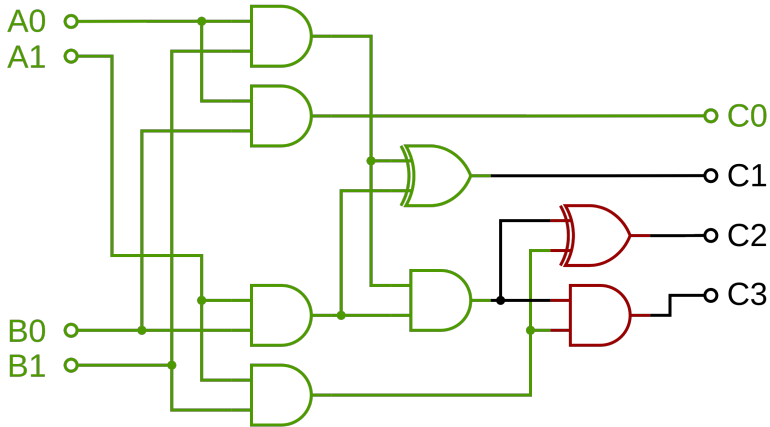
Propagation Delay vs Clock (but now the voltage is too low)



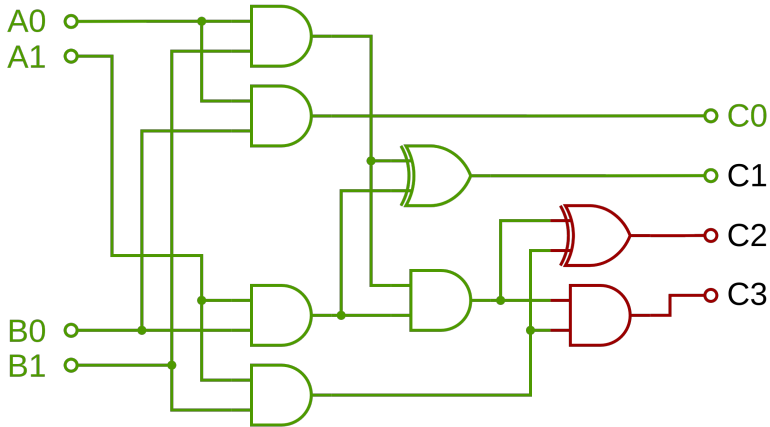
Propagation Delay vs Clock (but now the voltage is too low)



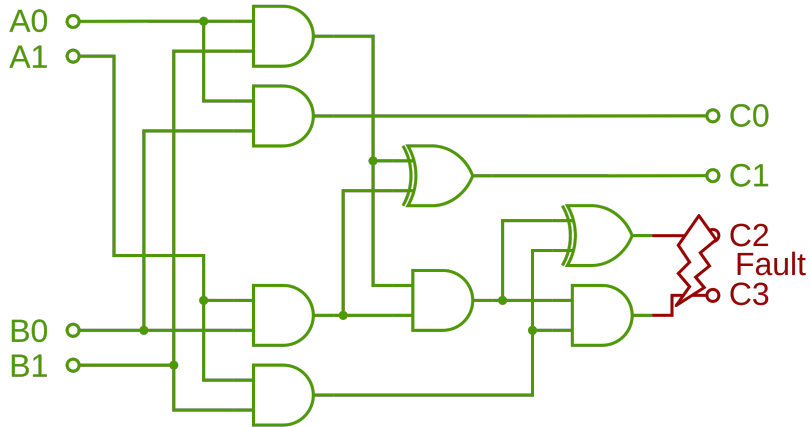
Propagation Delay vs Clock (but now the voltage is too low)



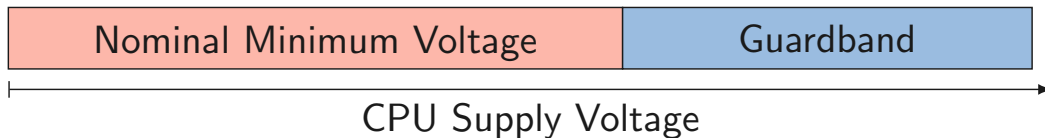
Propagation Delay vs Clock (but now the voltage is too low)



Propagation Delay vs Clock (but now the voltage is too low)



About Faulting Instructions cont.



Some Instruction Produce Faulty Results



Instruction	IMUL	VOR*	AESENC	VXOR*	VANDN*	VAND*	VSQRTPD	VPCLMULQDQ	VPSRAD	VPCMP*	VPMAX*	VPADDQ ¹
Number of Faults	79	47	40	40	30	28	24	16	9	5	3	1

¹A. Kogler, D. Gruss, and M. Schwarz. Minefield: A Software-only Protection for SGX Enclaves against DVFS Attacks. In: USENIX Security. 2022.

Exploiting Undervolting Faults

Clkscrew by Tang et al. [TSS17]

- ARM TrustZone
- Overclocking

Clkscrew by Tang et al. [TSS17]

- ARM TrustZone
- Overclocking

Voltage Jockey by Qiu et al. [Qiu+19]

- ARM TrustZone
- Undervolting

Clkscrew by Tang et al. [TSS17]

- ARM TrustZone
- Overclocking

Voltage Jockey by Qiu et al. [Qiu+19]

- ARM TrustZone
- Undervolting

Plundervolt by Murdock et al. [Mur+20]

- Intel SGX
- Undervolting

Attack Trusted Execution Environments

$$S = M^d \pmod{n}$$

$$S = M^d \pmod{n}$$

Chinese Remainder Theorem:

$$S_p = (M \bmod p)^{D \bmod p-1}$$

$$S_q = (M \bmod q)^{D \bmod q-1}$$

$$S = S_p \cdot (q^{-1} \bmod p) \cdot q + S_q \cdot (p^{-1} \bmod q) \cdot p \pmod{n}$$

Fault Attack on RSA Signatures with CRT (“Bellcore attack”)

- Compute signature twice and fault one computation ⚡

$$S = S_p \cdot (q^{-1} \bmod p) \cdot q + S_q \cdot (p^{-1} \bmod q) \cdot p \pmod{n}$$

$$S^{\text{⚡}} = S_p \cdot (q^{-1} \bmod p)^{\text{⚡}} \cdot q + S_q \cdot (p^{-1} \bmod q) \cdot p \pmod{n}$$

Fault Attack on RSA Signatures with CRT (“Bellcore attack”)

- Compute signature twice and fault one computation ⚡

$$\begin{aligned} S &= \boxed{S_p \cdot \text{something}} \cdot q + \boxed{S_q \cdot \text{some rest}} \cdot p \pmod{n} \\ S^{\text{⚡}} &= \boxed{S_p \cdot (q \text{ faulty mod } p)^q} \cdot q + \boxed{S_q \cdot \text{some rest}} \cdot p \pmod{n} \end{aligned}$$

Fault Attack on RSA Signatures with CRT (“Bellcore attack”)

- Compute signature twice and fault one computation ⚡

$$\begin{aligned} S &= \boxed{\text{something}} \cdot q + \boxed{\text{some rest}} \pmod{n} \\ S^{\text{⚡}} &= \boxed{S_p \cdot (\text{faulty} \pmod p)^q} \cdot q + \boxed{\text{some rest}} \pmod{n} \\ S - S^{\text{⚡}} &= \boxed{\text{some garbage}} \cdot q + \boxed{0} \pmod{n} \end{aligned}$$

Fault Attack on RSA Signatures with CRT (“Bellcore attack”)

- Compute signature twice and fault one computation ⚡

$$\begin{aligned} S &= \boxed{\text{something}} \cdot q + \boxed{\text{some rest}} \pmod{n} \\ S^\text{⚡} &= \boxed{S_p \cdot (q \bmod p)^\text{⚡}} \cdot q + \boxed{\text{some rest}} \pmod{n} \\ S - S^\text{⚡} &= \boxed{\text{some garbage}} \cdot q + \boxed{0} \pmod{n} \end{aligned}$$

- Get the secret q using

$$\gcd(S - S^\text{⚡}, n) = \gcd(\boxed{\text{some garbage}} \cdot q, p \cdot q) = q$$

Demo

**But undervolting could save so
much power...**

SUIT

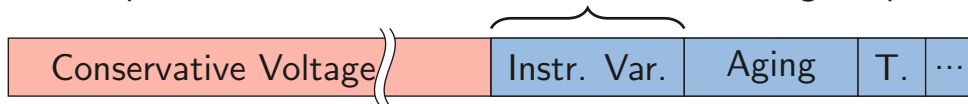
Secure Undervolting with Instruction Traps

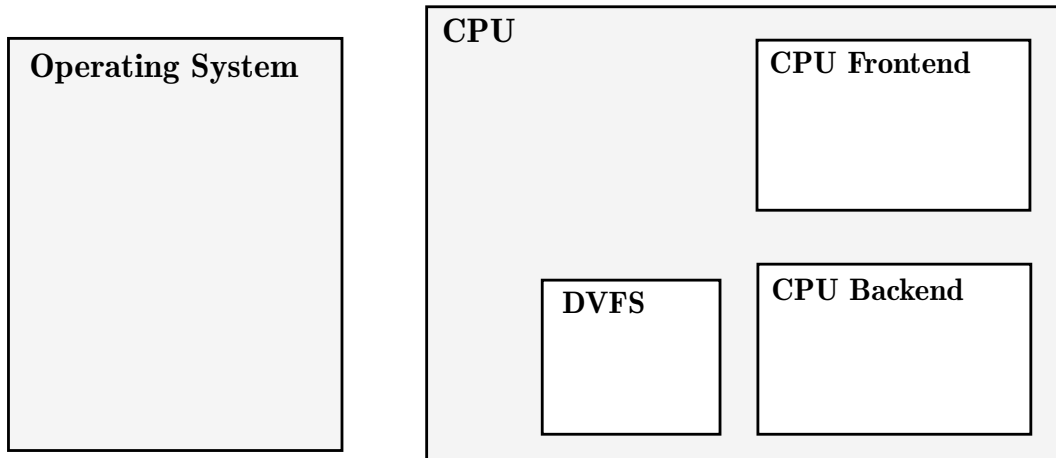
Jonas Juffinger, Stepan Kalinin, Daniel Gruss, Frank Müller

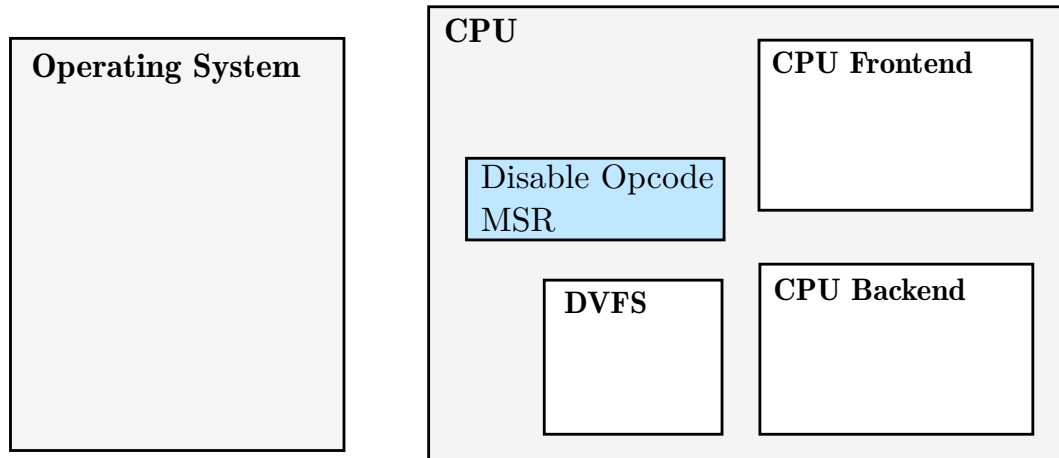
ASPLOS 2024, San Diego, USA — April 27- May 1, 2024

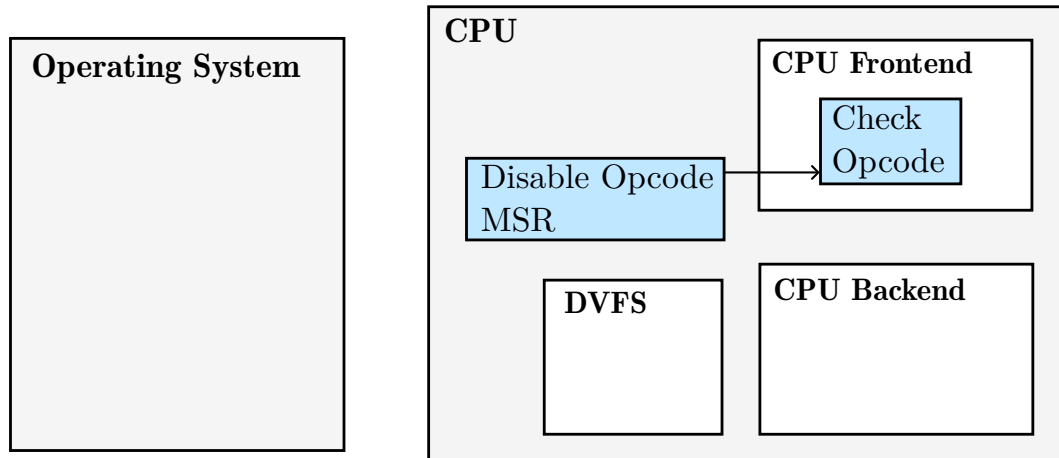


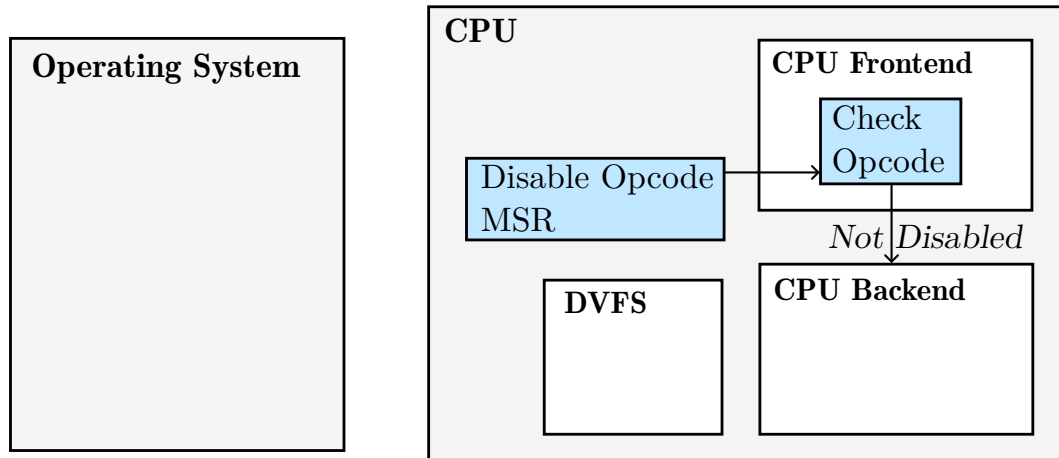
Up to a 150 mV variation in instruction voltage requirement.

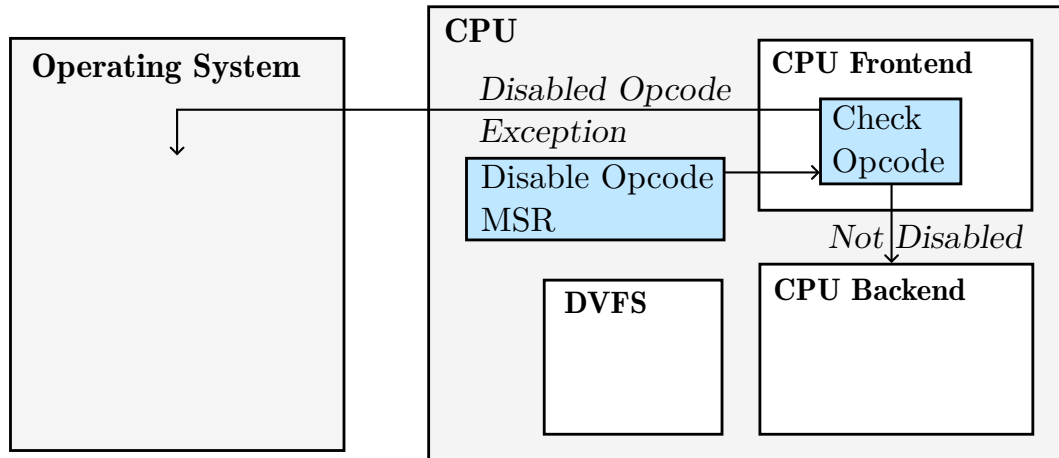


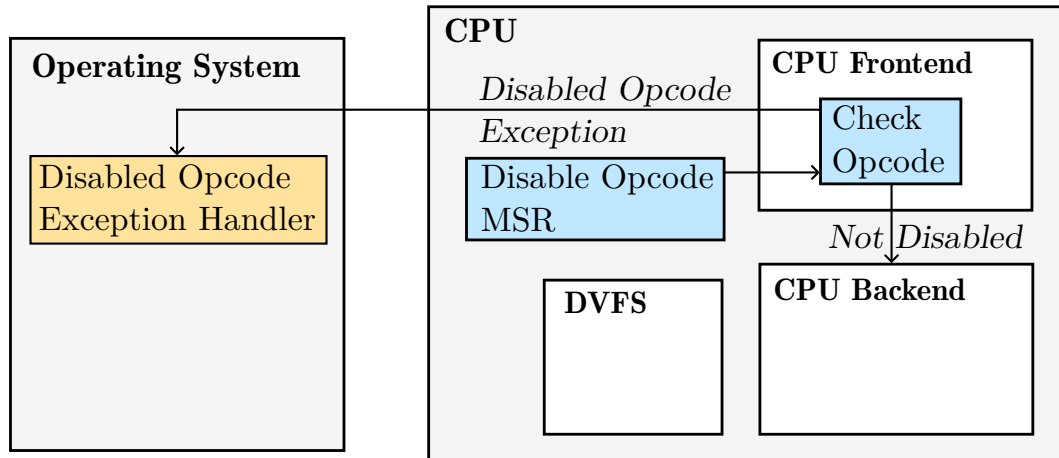


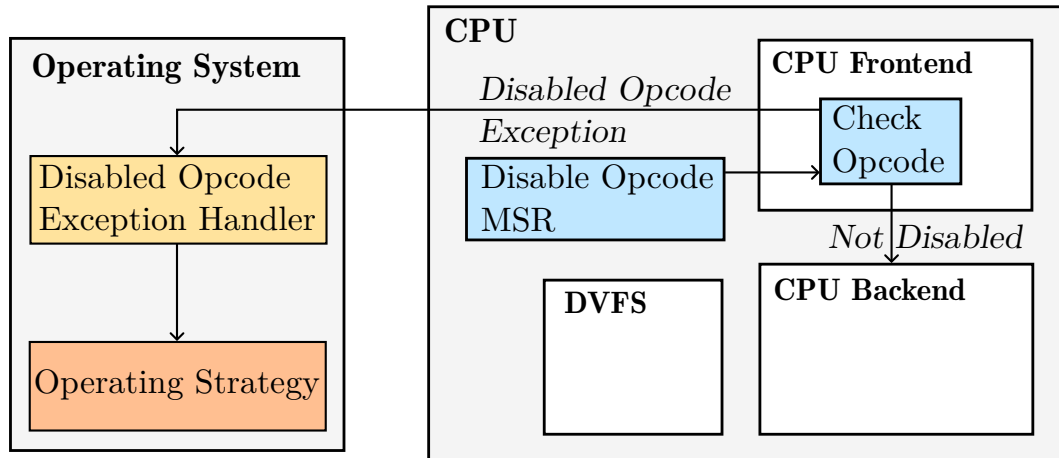


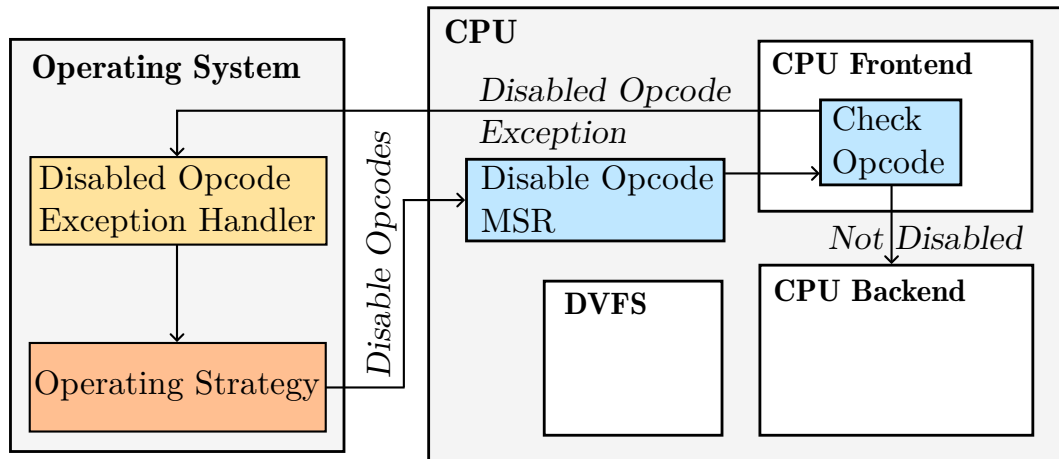


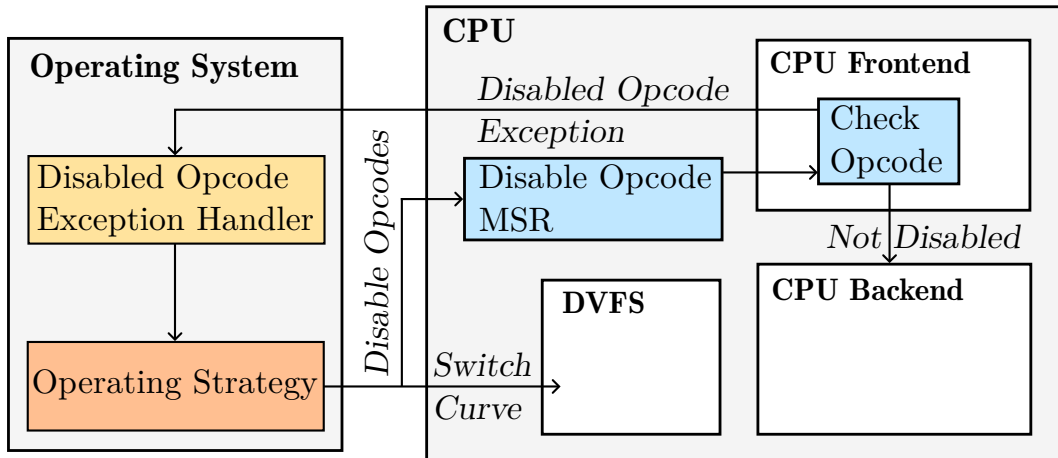


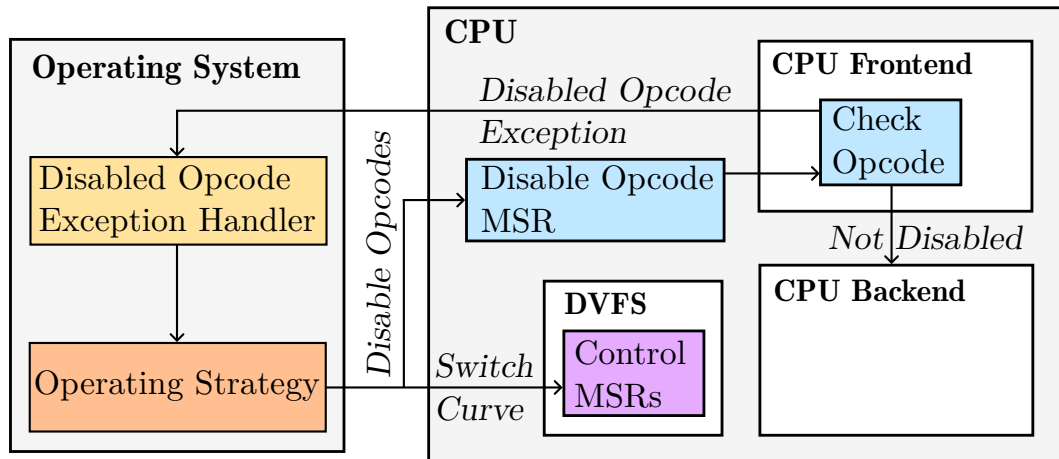


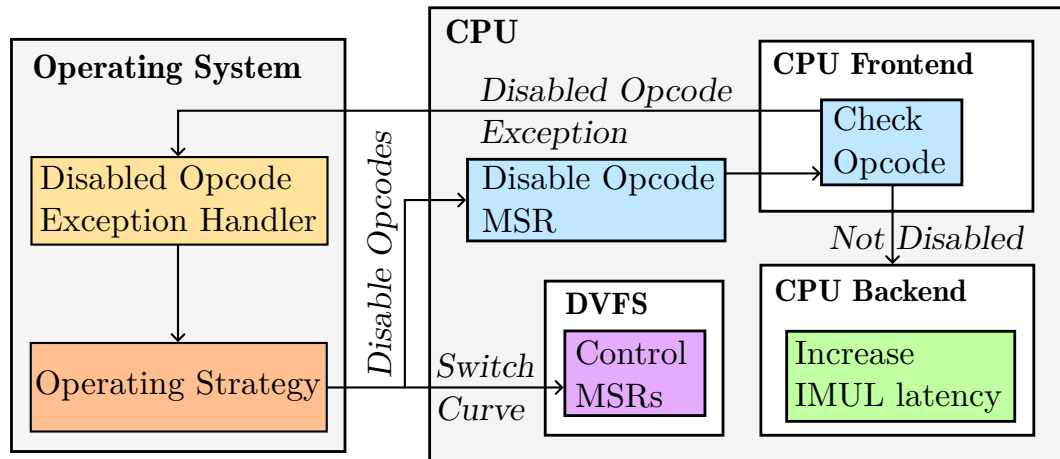


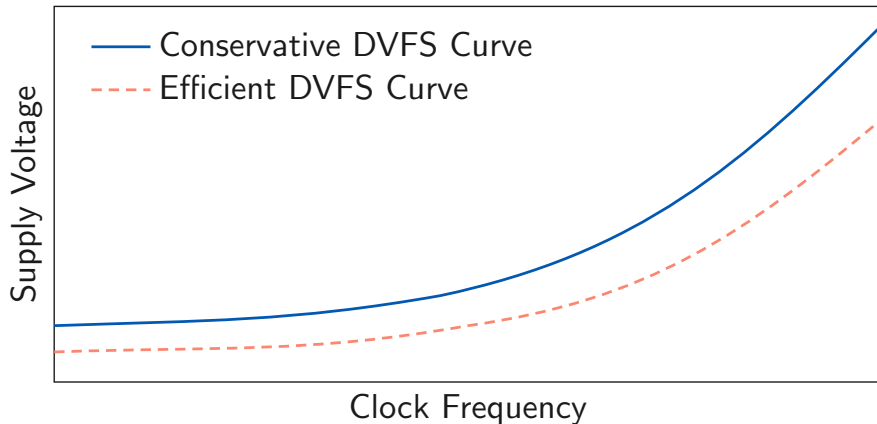


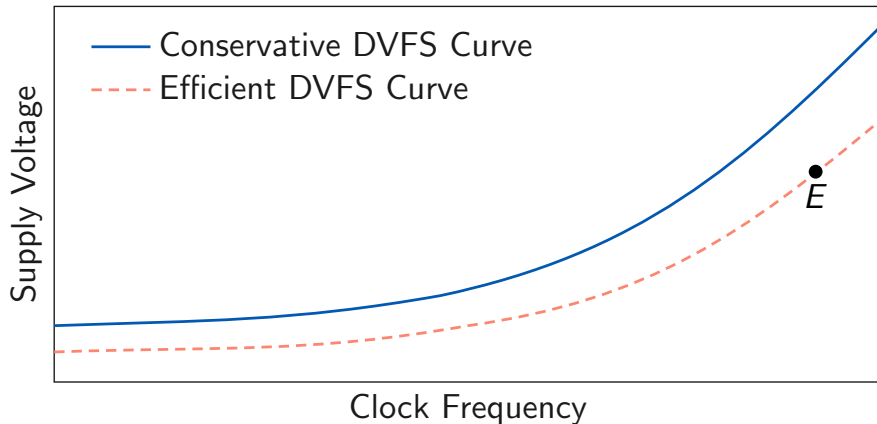


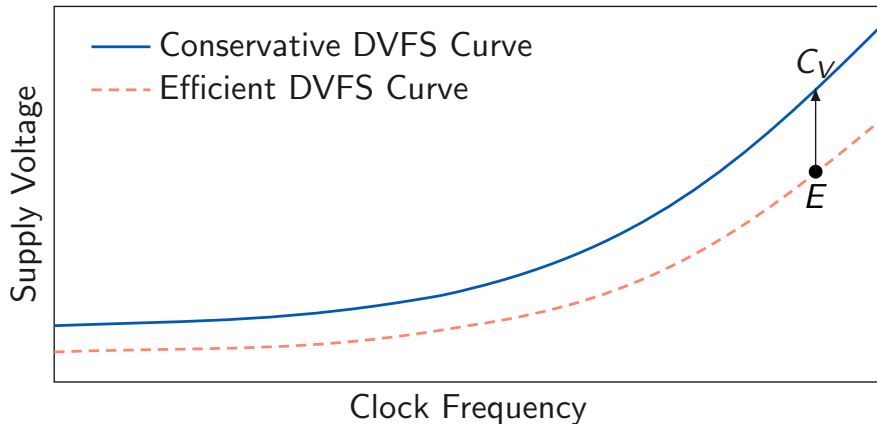


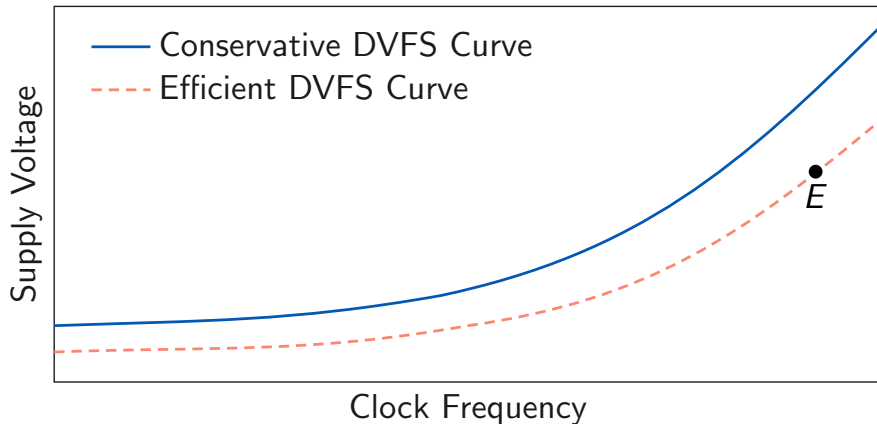


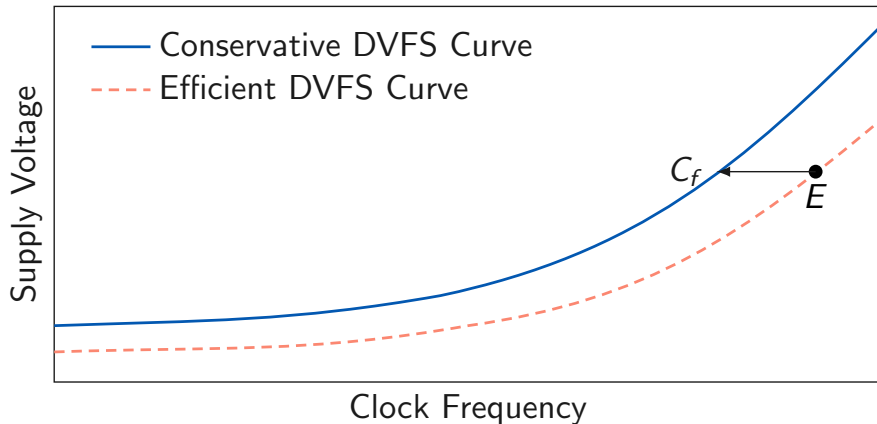


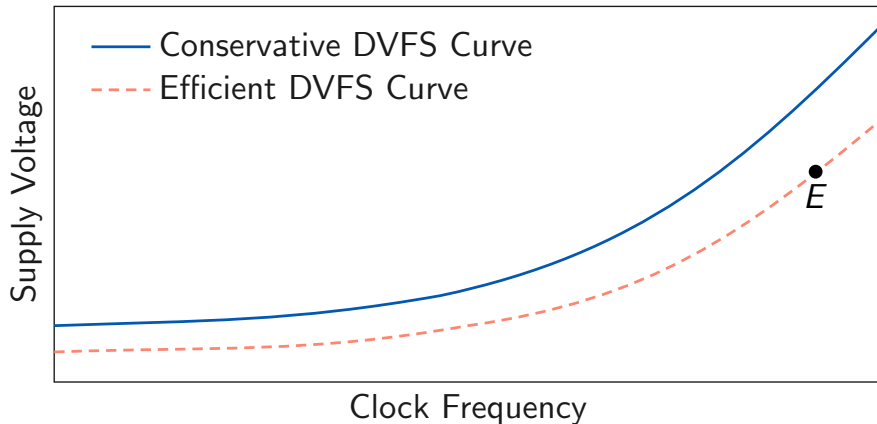


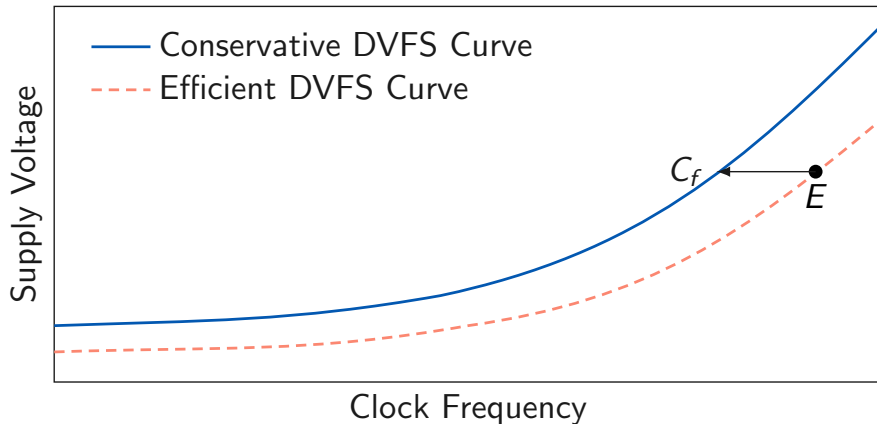


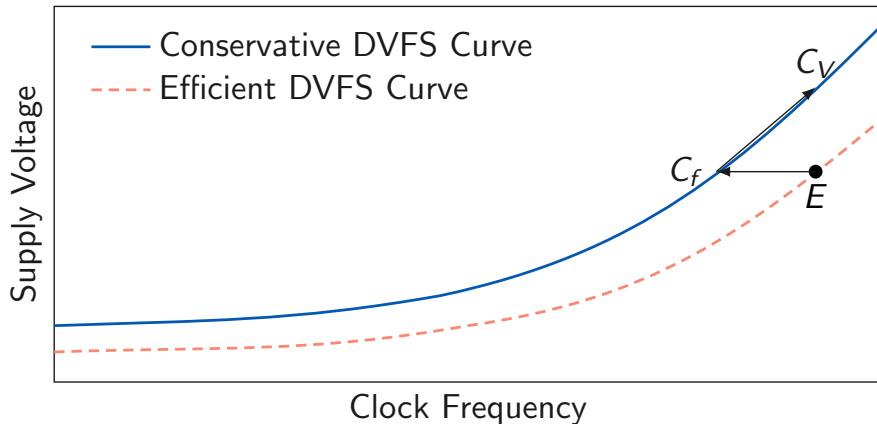


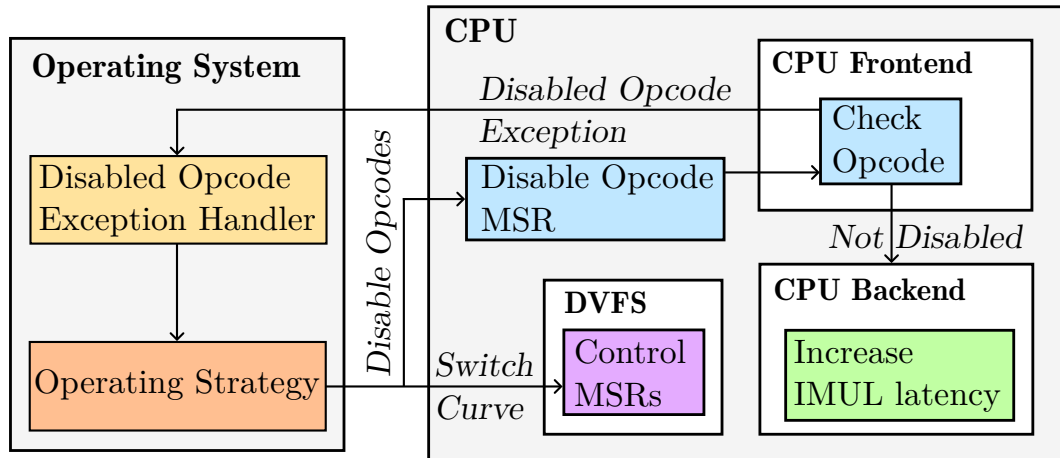


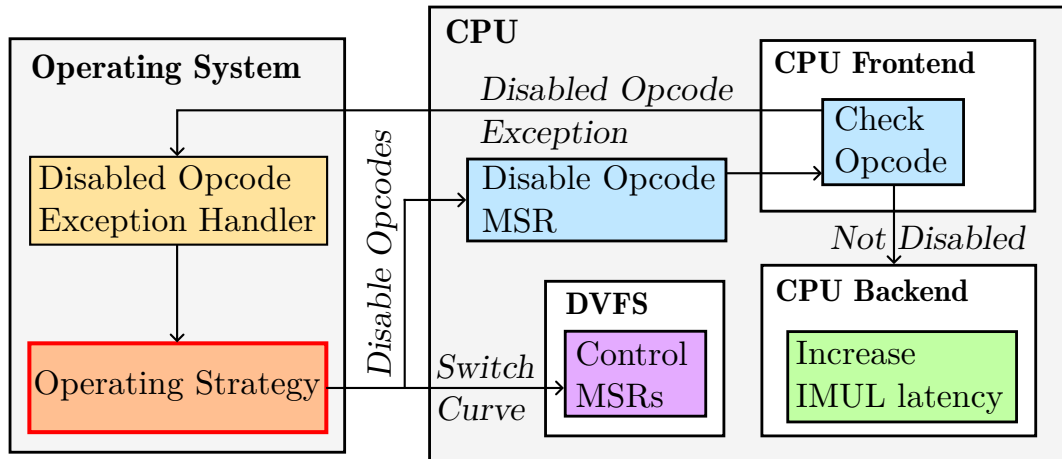


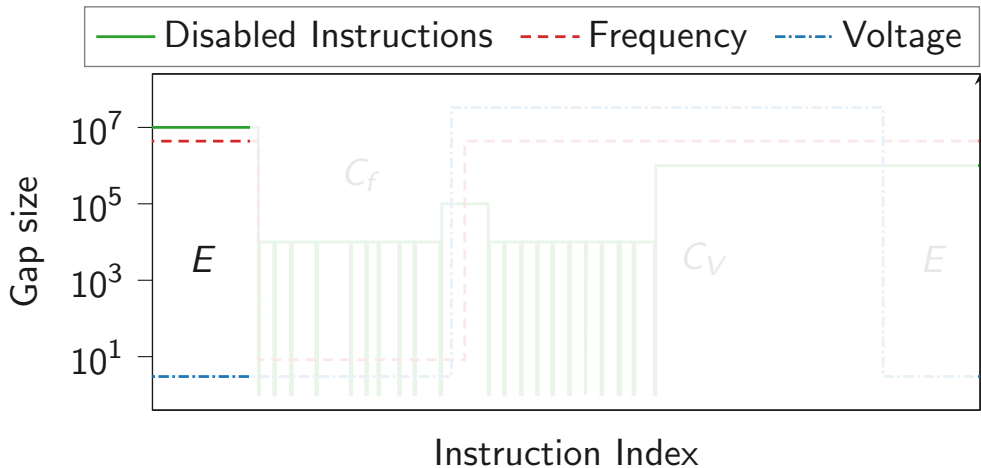


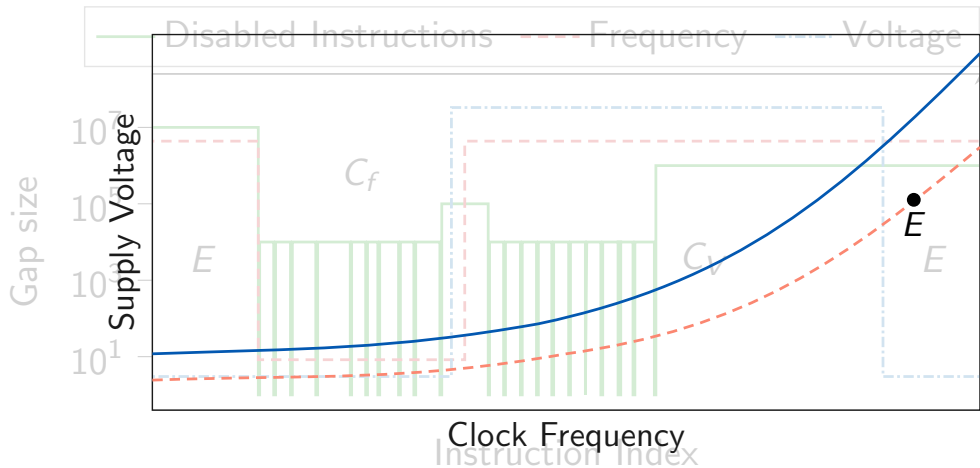


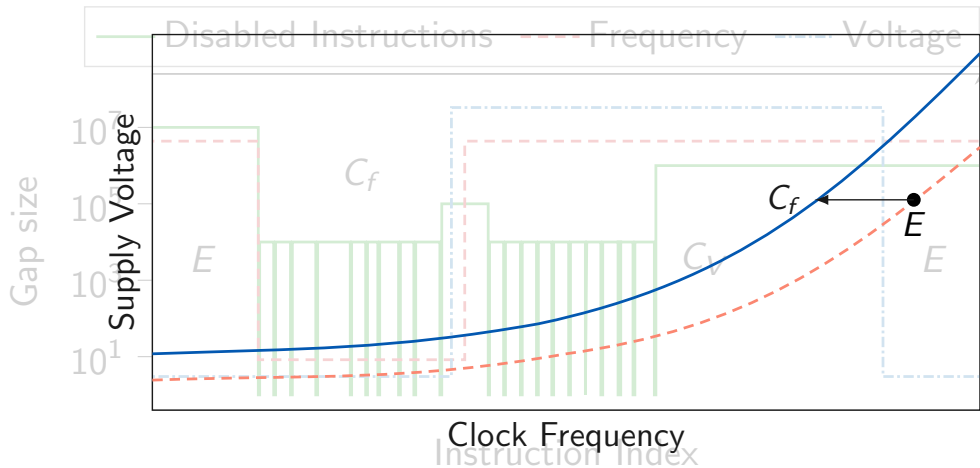


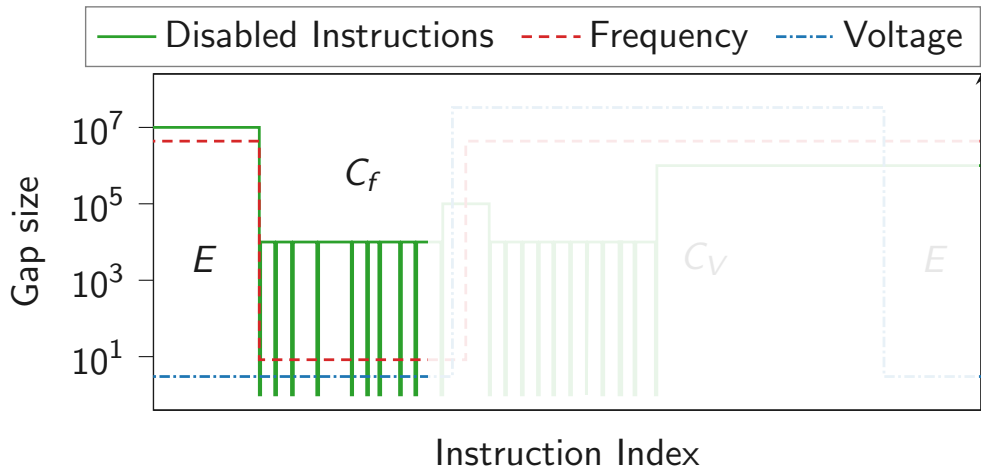


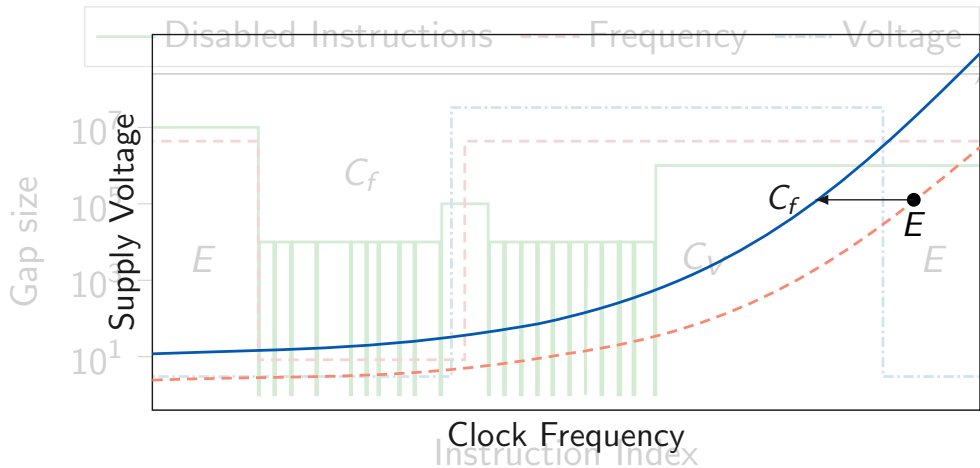


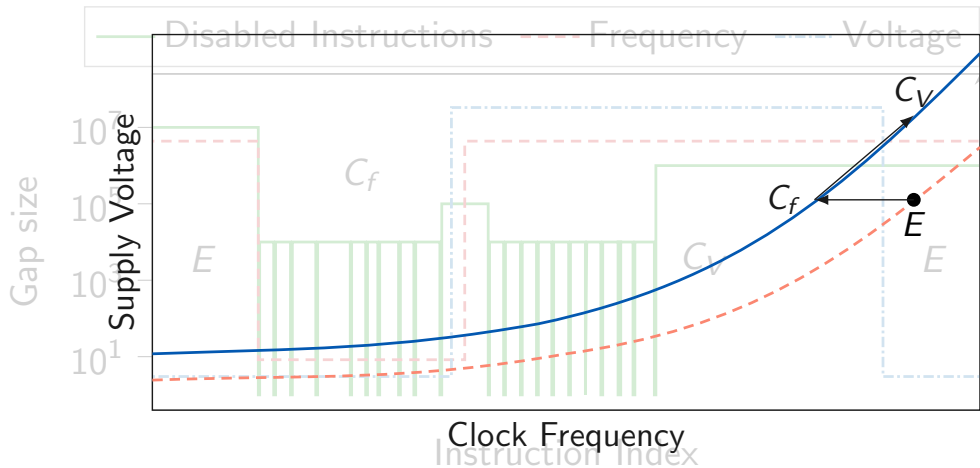


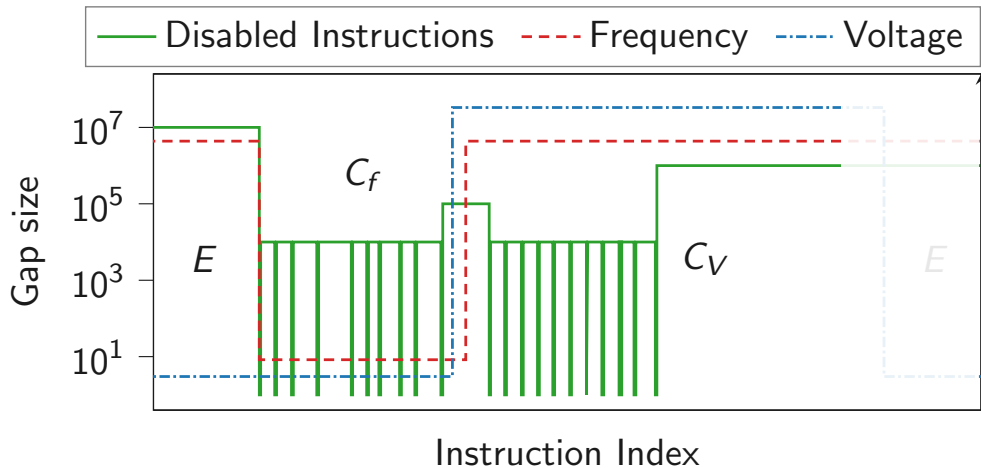


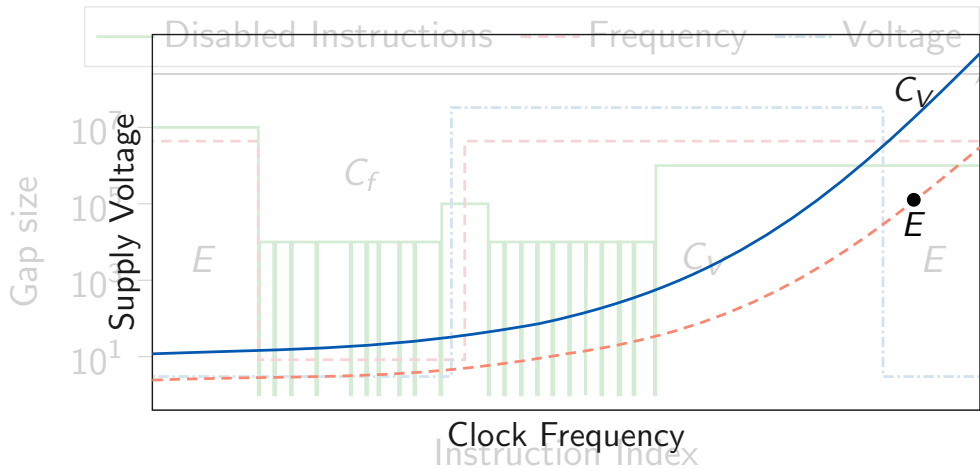


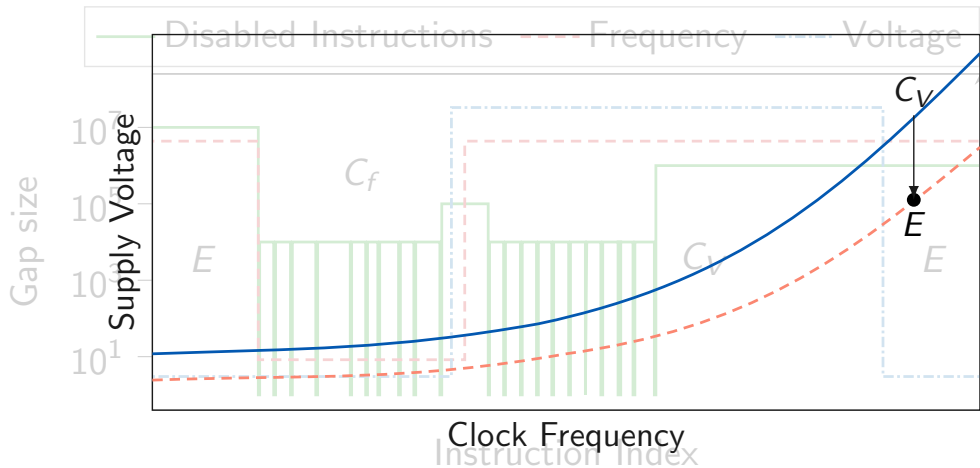


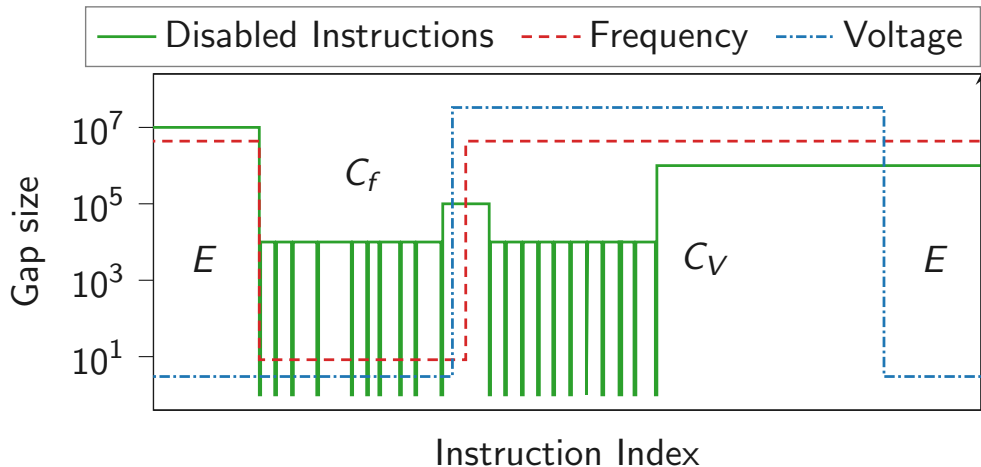


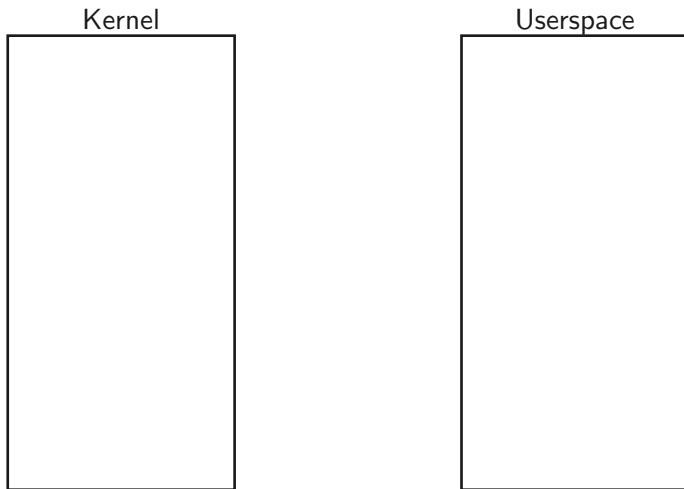


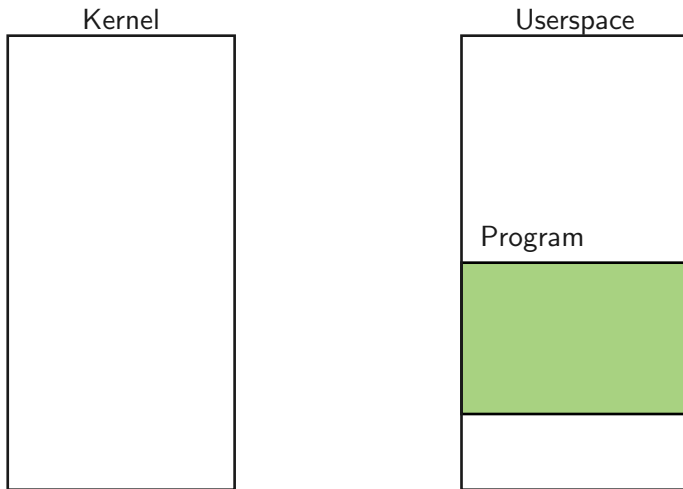


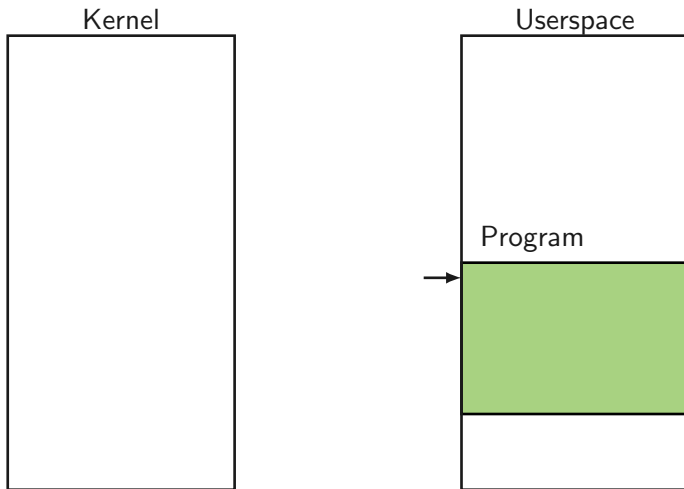


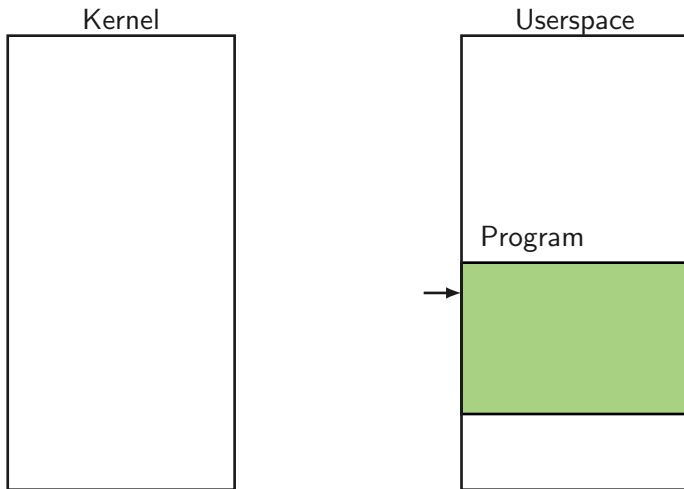


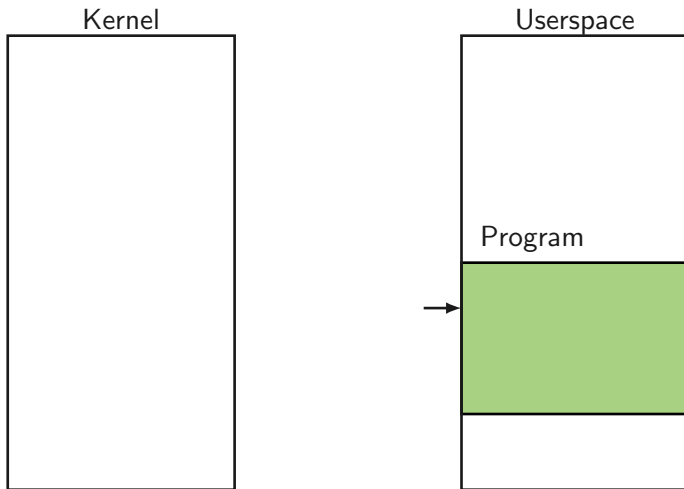


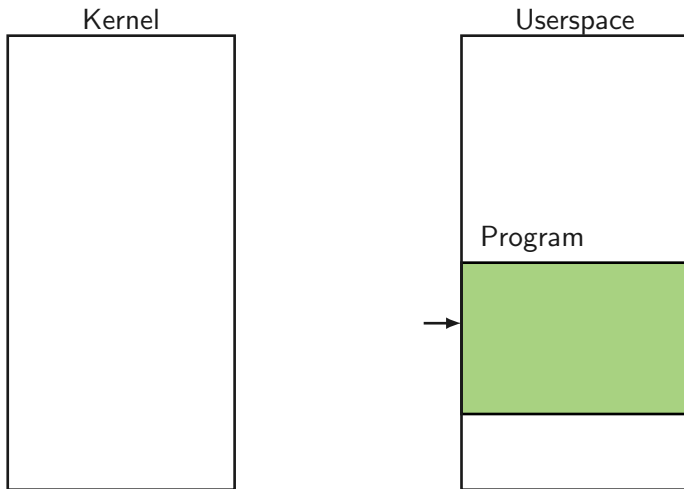


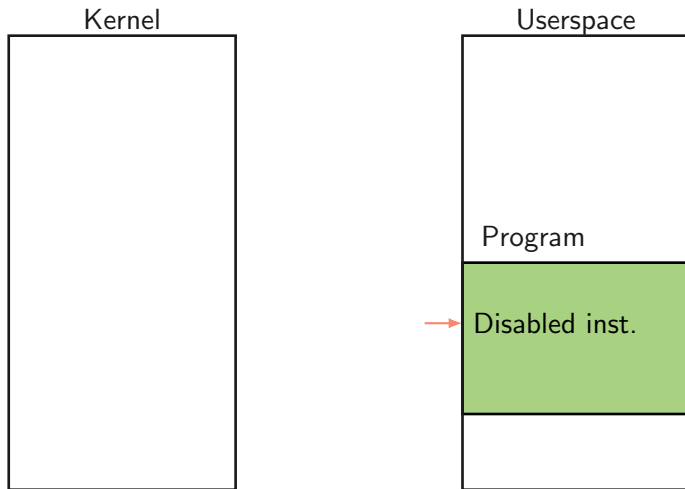


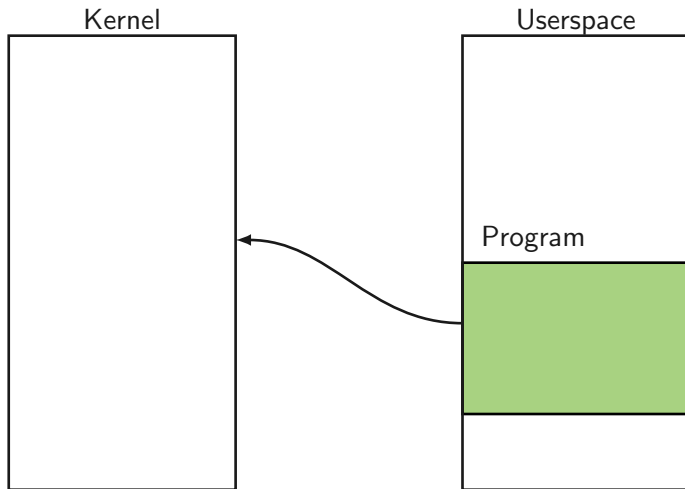


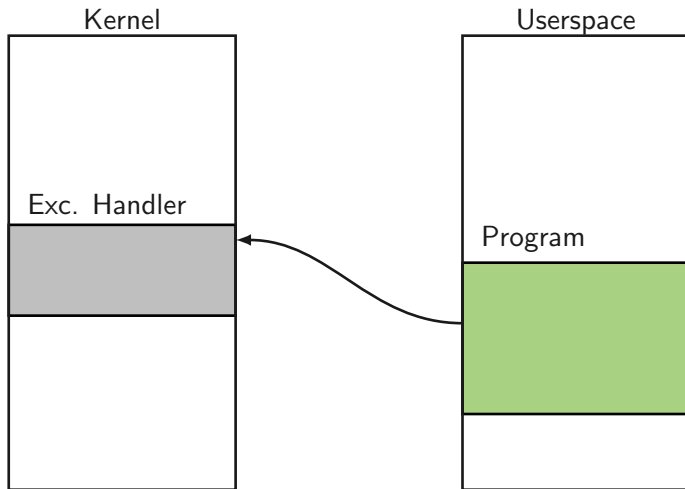


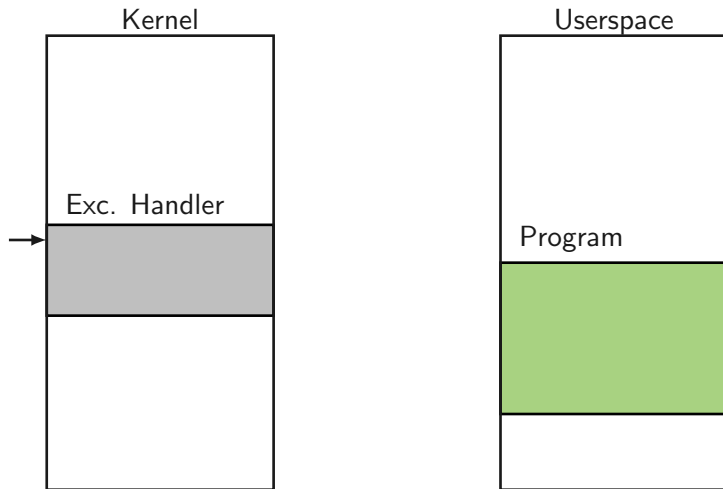


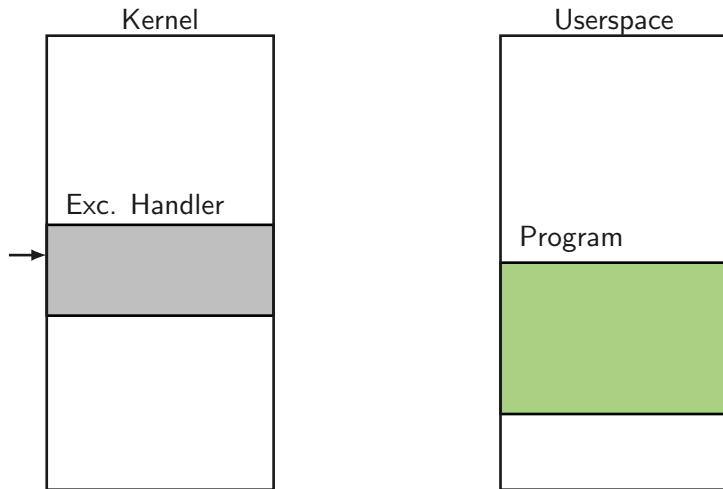


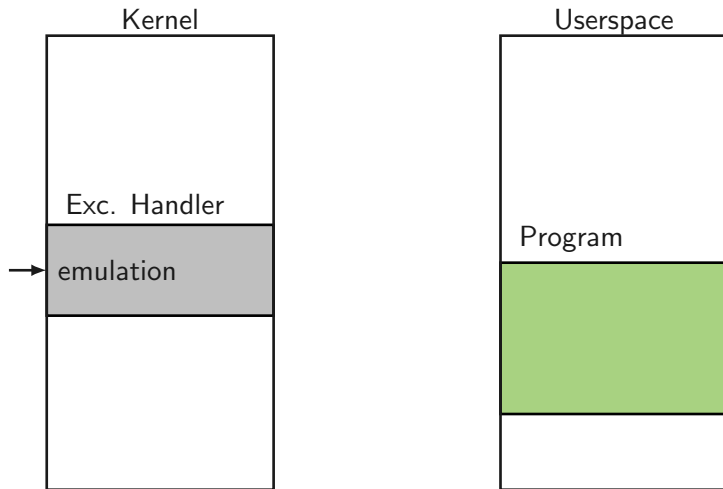


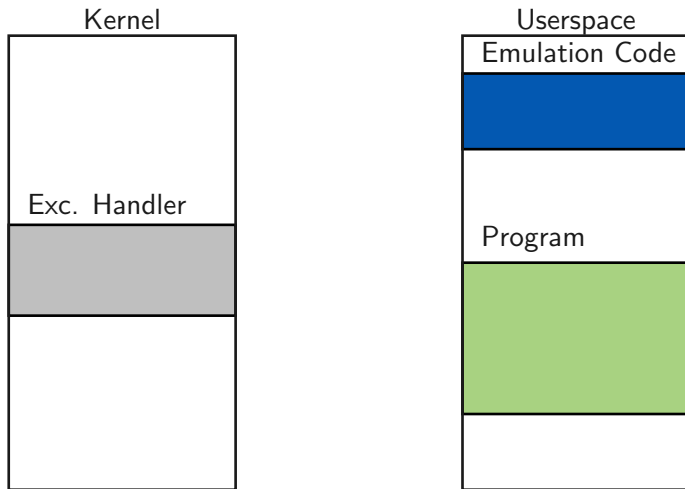


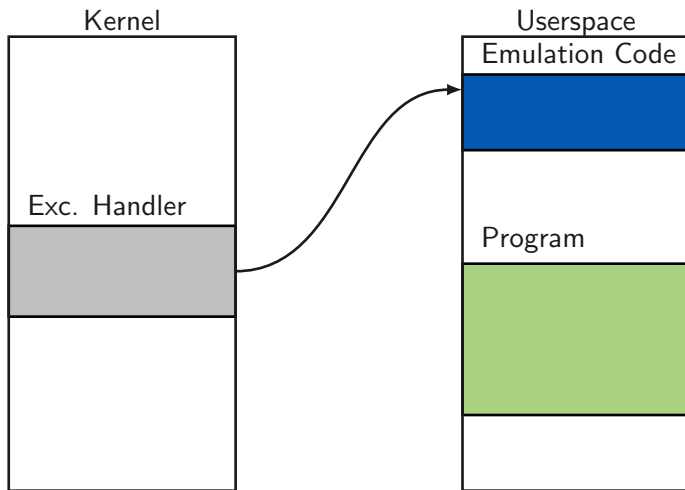


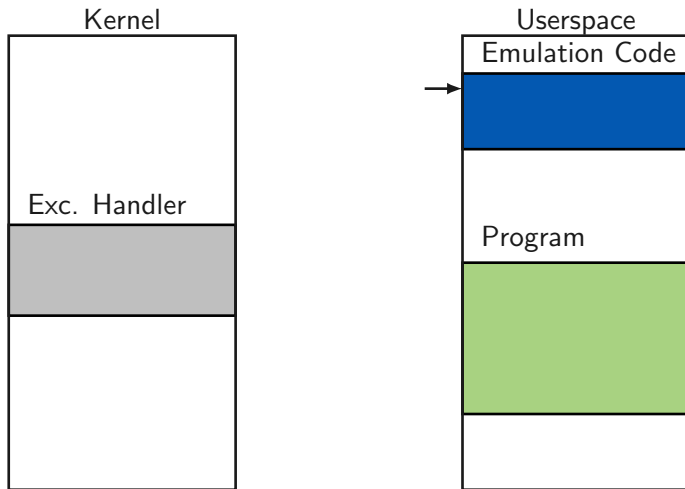


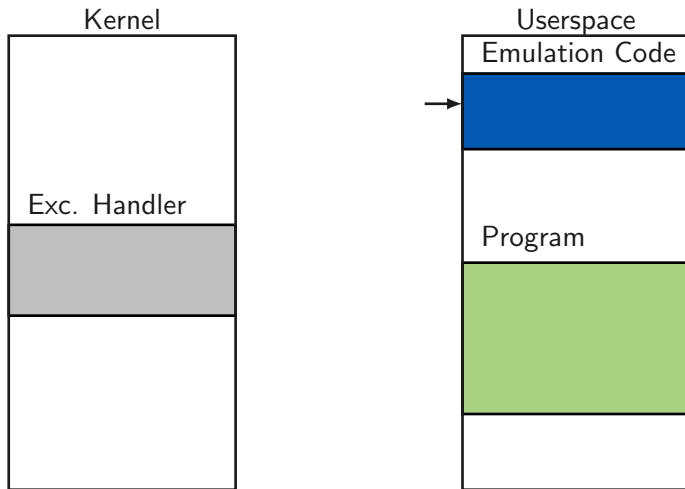


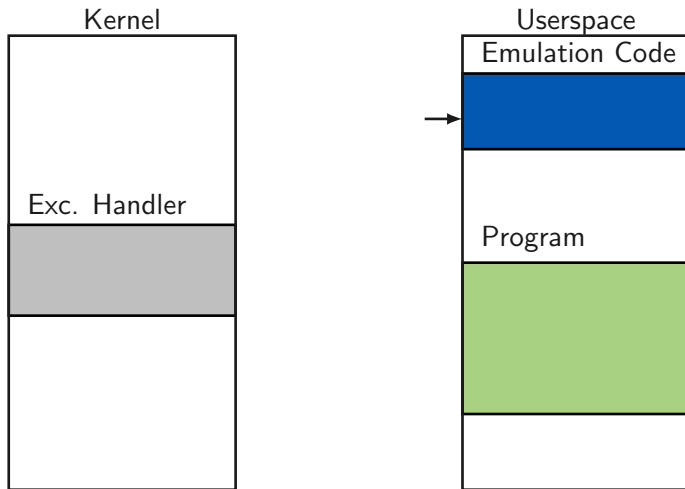


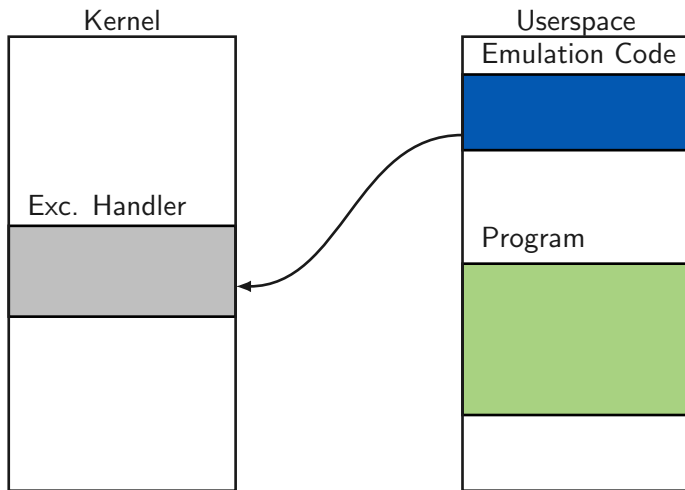


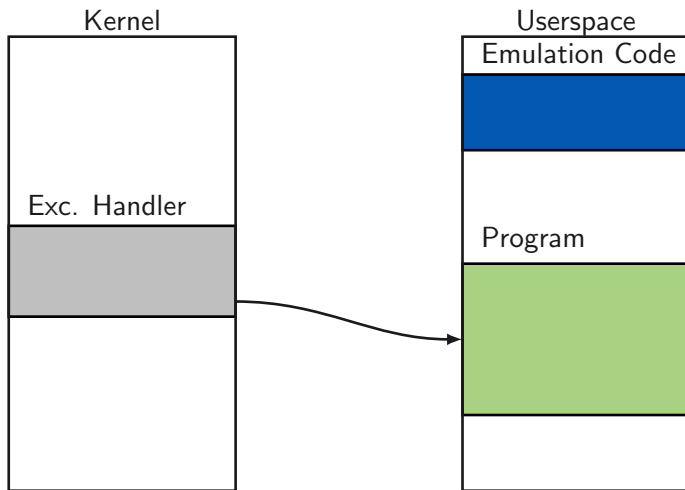


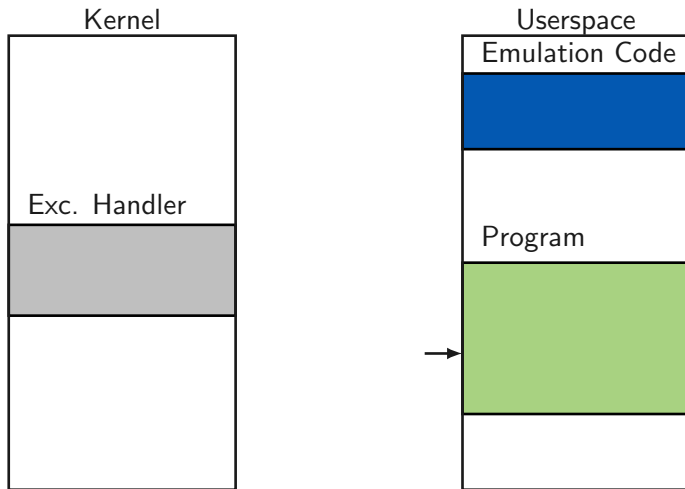


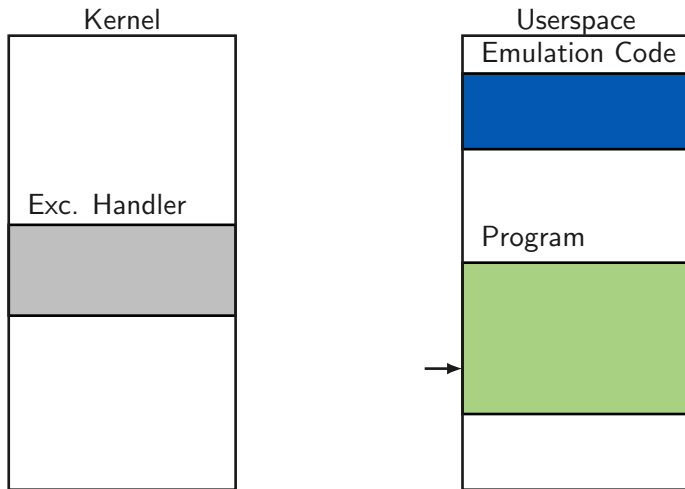


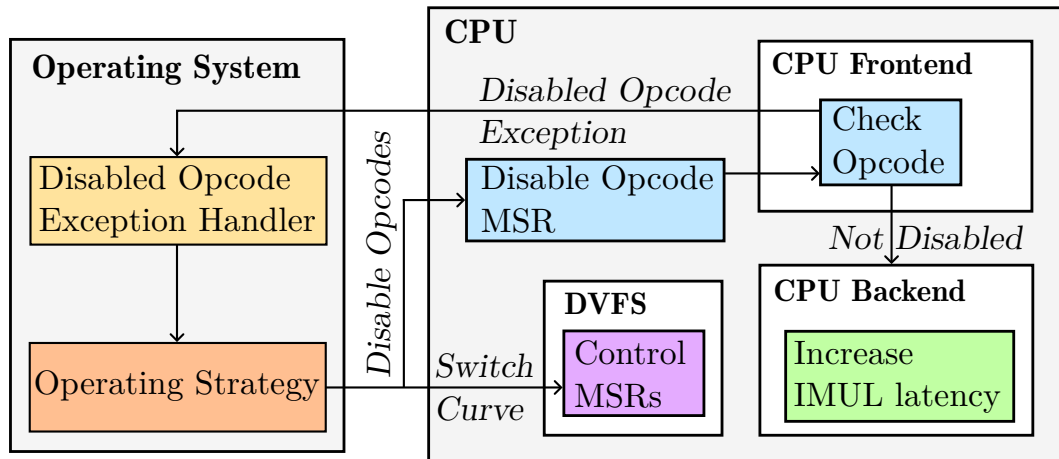


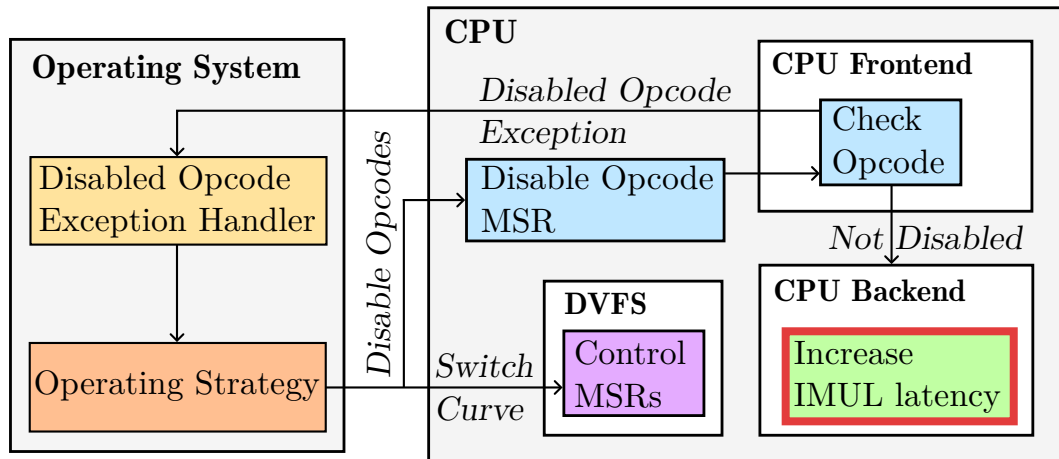




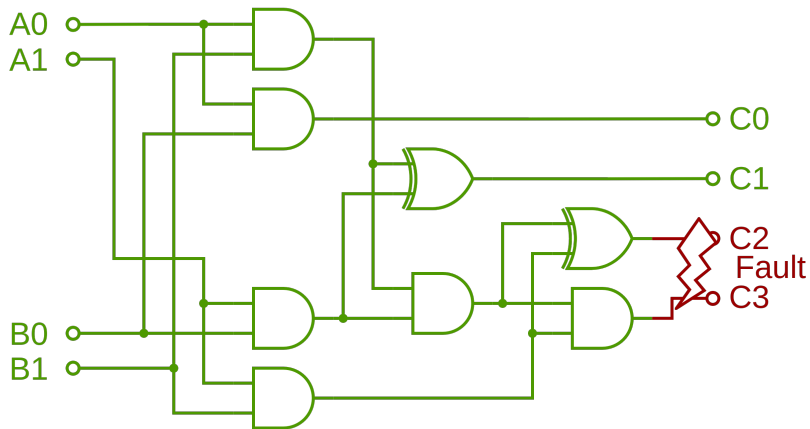


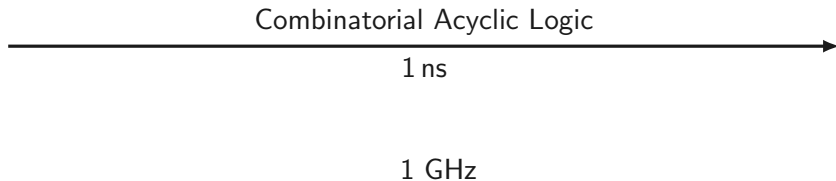




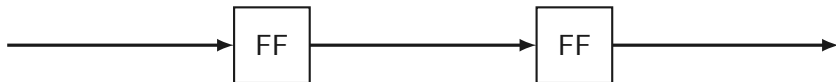


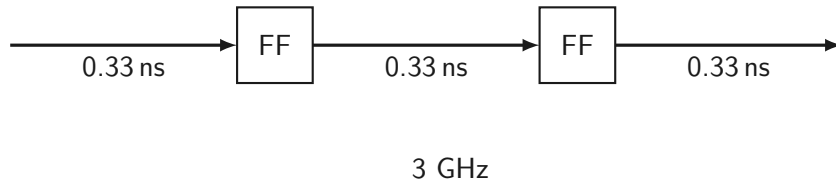
Increase IMUL Latency



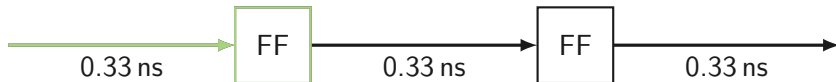




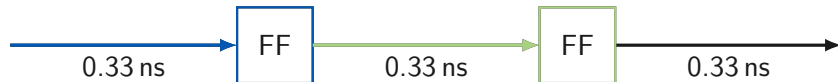




Increase IMUL Latency



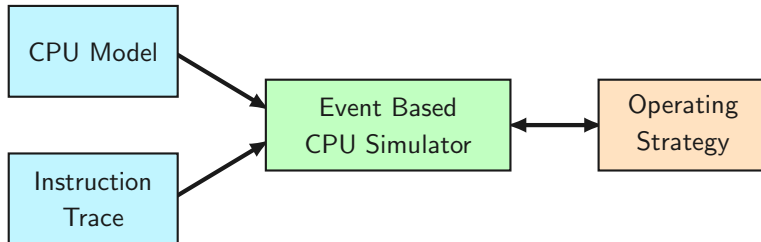
Increase IMUL Latency



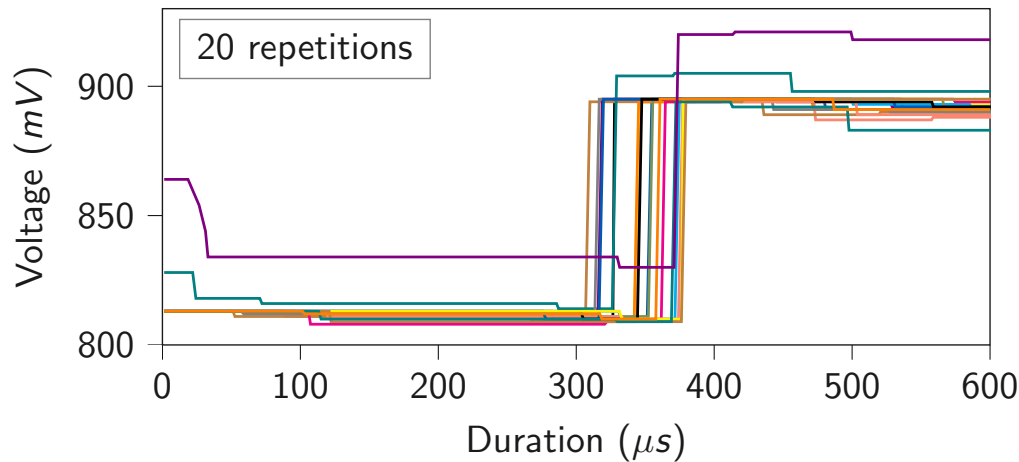


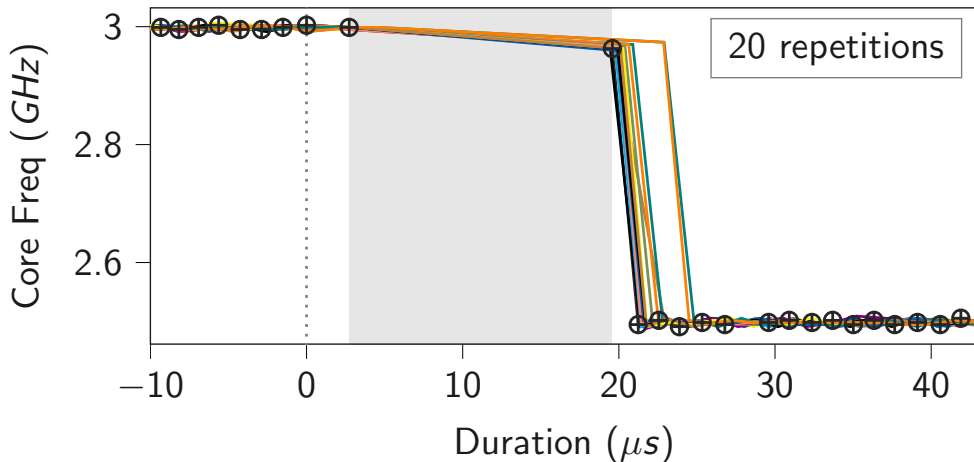
Latency: 3, Throughput: 1

Evaluation



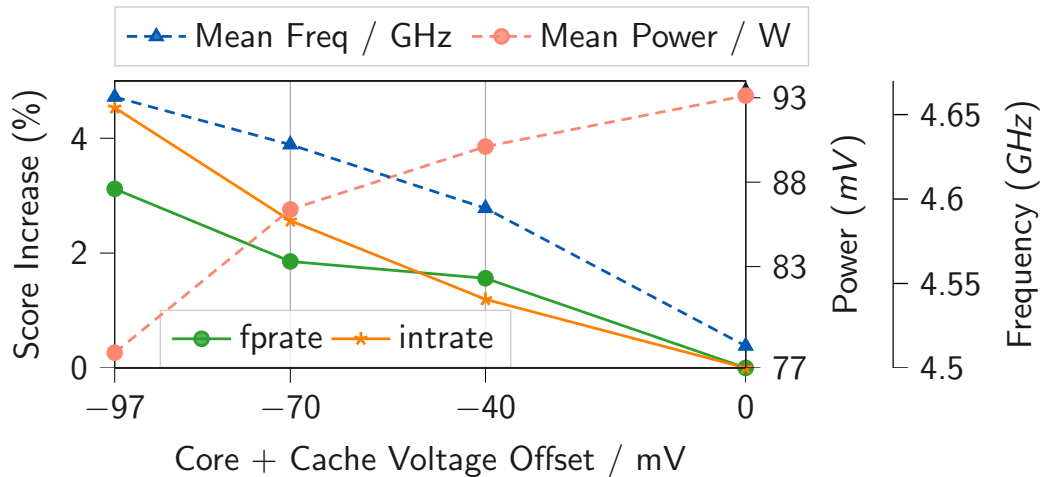
Voltage Change Delay



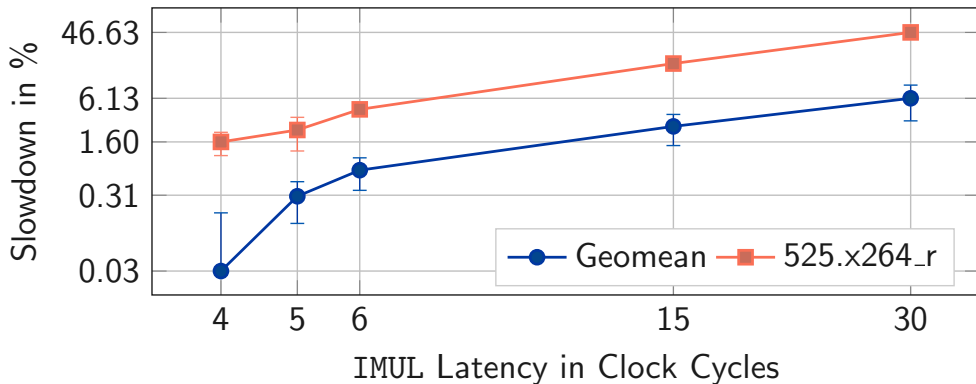


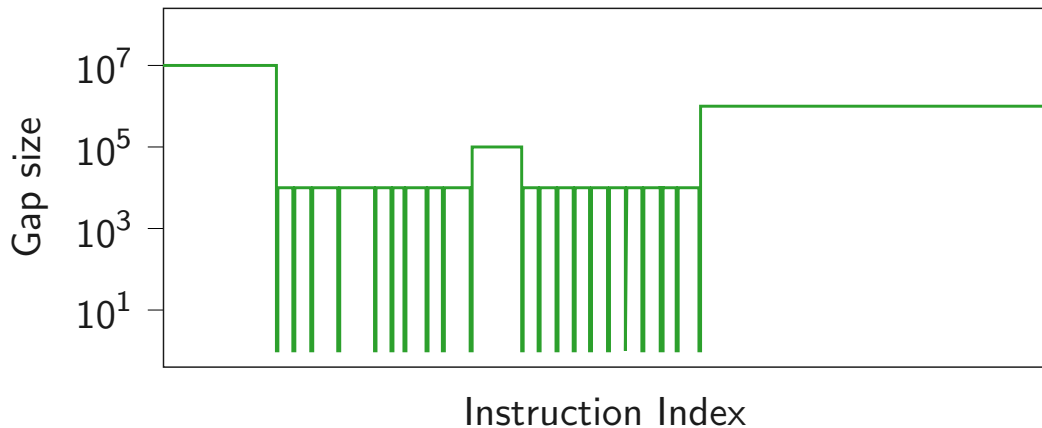
CPU	V_{off}	Score	Power	Freq.	Eff.
i5-1035G1	-70 mV	+6.0 %	-0.1 %	+8.5 %	+6.1 %
	-97 mV	+7.9 %	-0.5 %	+12 %	+8.4 %
i9-9900K	-70 mV	+2.2 %	-7.2 %	+2.6 %	+10 %
	-97 mV	+3.8 %	-16 %	+3.3 %	+23 %
7700X*	-70 mV	+1.4 %	-9.8 %	+1.8 %	+12 %
	-97 mV	+1.9 %	-15 %	+1.8 %	+20 %

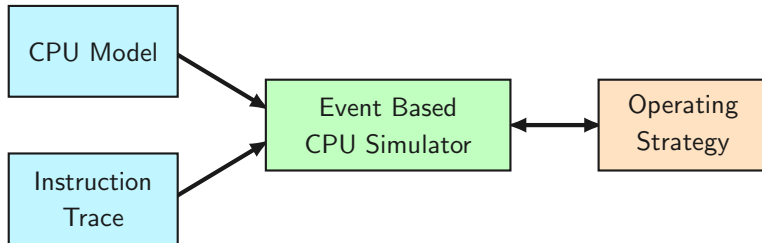
Performance Improvement and Power Savings (as a graph)



Increasing the IMUL Latency







```

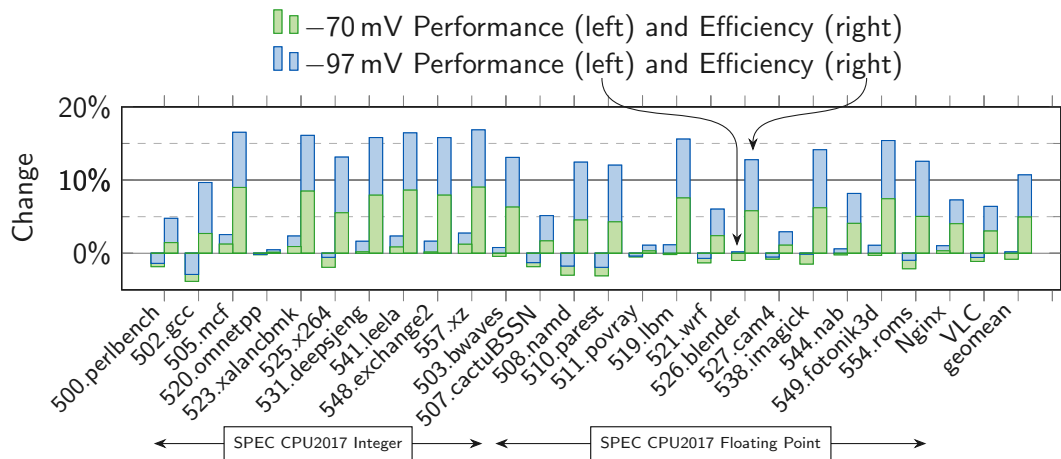
class Operating_Strategy_fV:
    def disabled_instruction_exception_handler():
        # we wait for the frequency to change
        cpu.change_pstate_wait(DVFS.Cf)
        # and request the voltage change
        cpu.change_pstate_async(DVFS.Cv)

        cpu.set_instructions_disabled(False)

        # trashing prevention
        if exception_count_in_timespan(p_ts) >= p_ec:
            cpu.set_timer_interrupt(p_dl * p_df)
        else:
            cpu.set_timer_interrupt(p_dl)

    def timer_interrupt_handler():
        cpu.set_instructions_disabled(True)
        cpu.change_pstate_async(DVFS.E)

```



More Results

		70 mV Undervolt							97 mV Undervolt					
CPU _{cores}	OS		SPEC _{gmean}	SPEC _{median}	525.x264	SPEC _{noSIMD}	Nginx	VLC	SPEC _{gmean}	SPEC _{median}	525.x264	SPEC _{noSIMD}	Nginx	VLC
A ₁	fV	Pwr	-5.6 %	-7.1 %	-7.1 %	-7.1 %	-3.5 %	-3.9 %	-9.7 %	-11 %	-12 %	-15 %	-5.8 %	-6.3 %
		Perf.	-0.2 %	-1.3 %	-1.3 %	+3.0 %	+0.5 %	-0.4 %	+0.8 %	+1.3 %	0.1 %	+3.4 %	+1.2 %	+0.2 %
		Eff.	+5.7 %	+6.2 %	+6.2 %	+11 %	+4.2 %	+3.6 %	+12 %	+14 %	+14 %	+21 %	+7.4 %	+6.9 %
A ₄	fV	Pwr	-4.6 %	-0.1 %	-6.9 %	-7.4 %	-1.0 %	-1.0 %	-8.9 %	-8.7 %	-13 %	-16 %	-1.6 %	-1.6 %
		Perf.	-3.9 %	-0.0 %	-7.9 %	+1.8 %	-0.3 %	-0.6 %	-3.6 %	-3.5 %	-7.2 %	+1.8 %	-0.1 %	-0.5 %
		Eff.	+0.7 %	0.1 %	-1.0 %	+10.0 %	+0.7 %	+0.4 %	+5.8 %	+5.7 %	+6.7 %	+22 %	+1.5 %	+1.1 %
A _∞	e	Pwr	-7.5 %	-7.6 %	-5.4 %	-7.5 %	-7.2 %	-7.2 %	-12 %	-12 %	-10 %	-17 %	-12 %	-12 %
		Perf.	-42 %	-12 %	+6.2 %	+1.4 %	-98 %	-92 %	-42 %	-12 %	+6.1 %	+1.4 %	-98 %	-92 %
		Eff.	-37 %	-4.5 %	+12 %	+9.6 %	-98 %	-91 %	-34 %	+0.6 %	+18 %	+22 %	-98 %	-91 %
B _∞	f	Pwr	-8.1 %	-7.8 %	-7.8 %	-9.1 %	-4.4 %	-4.4 %	-12 %	-11 %	-11 %	-14 %	-6.7 %	-6.7 %
		Perf.	-7.8 %	-7.8 %	-9.2 %	+0.4 %	-2.5 %	-2.5 %	-10 %	-11 %	-12 %	+0.6 %	-2.3 %	-2.3 %
		Eff.	+0.3 %	-0.0 %	-1.6 %	+11 %	+2.0 %	+2.0 %	+1.4 %	0.1 %	-1.6 %	+17 %	+4.7 %	+4.7 %
	e	Pwr	-9.2 %	-8.0 %	-11 %	-9.2 %	-9.8 %	-9.8 %	-14 %	-13 %	-16 %	-14 %	-15 %	-15 %
		Perf.	-26 %	-5.1 %	+15 %	-0.5 %	-96 %	-80 %	-26 %	-5.2 %	+19 %	0.0 %	-96 %	-80 %
		Eff.	-19 %	+3.1 %	+28 %	+9.5 %	-95 %	-78 %	-14 %	+9.3 %	+41 %	+17 %	-95 %	-76 %
C _∞	fV	Pwr	-5.6 %	-7.1 %	-7.1 %	-6.1 %	-3.6 %	-4.0 %	-9.8 %	-11 %	-12 %	-14 %	-5.8 %	-6.6 %
		Perf.	-0.8 %	-1.9 %	-1.9 %	+3.5 %	+0.3 %	-1.1 %	+0.2 %	+0.2 %	-0.6 %	+3.8 %	+1.0 %	-0.6 %
		Eff.	+5.1 %	+5.5 %	+5.5 %	+10 %	+4.0 %	+3.0 %	+11 %	+13 %	+13 %	+21 %	+7.3 %	+6.4 %



Undervolting can made secure



Undervolting can made secure
If you want to learn more about:



Undervolting can made secure

If you want to learn more about:

- Analysis of aging and temperature guardband
- Exception delay measurement
- Not using faulting instruction
- Detailed security analysis



Undervolting can be made secure

If you want to learn more about:

- Analysis of aging and temperature guardband
- Exception detection measurement
- Not using the instruction
- Detailed security analysis

Read the Paper



SUIT

Secure Undervolting with Instruction Traps

Jonas Juffinger, Stepan Kalinin, Daniel Gruss, Frank Müller

ASPLOS 2024, San Diego, USA — April 27- May 1, 2024

✉ jonas.juffinger@iaik.tugraz.at [@notimaginary_](https://twitter.com/notimaginary_) www.jonasjuffinger.com

- [KGS22] A. Kogler, D. Gruss, and M. Schwarz. Minefield: A Software-only Protection for SGX Enclaves against DVFS Attacks. In: USENIX Security. 2022.
- [Mur+20] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In: S&P. 2020.
- [Qiu+19] P. Qiu, D. Wang, Y. Lyu, and G. Qu. VoltJockey: Breaking SGX by Software-Controlled Voltage-Induced Hardware Faults. In: AsianHOST. 2019.
- [TSS17] A. Tang, S. Sethumadhavan, and S. Stolfo. CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management. In: USENIX Security. 2017.