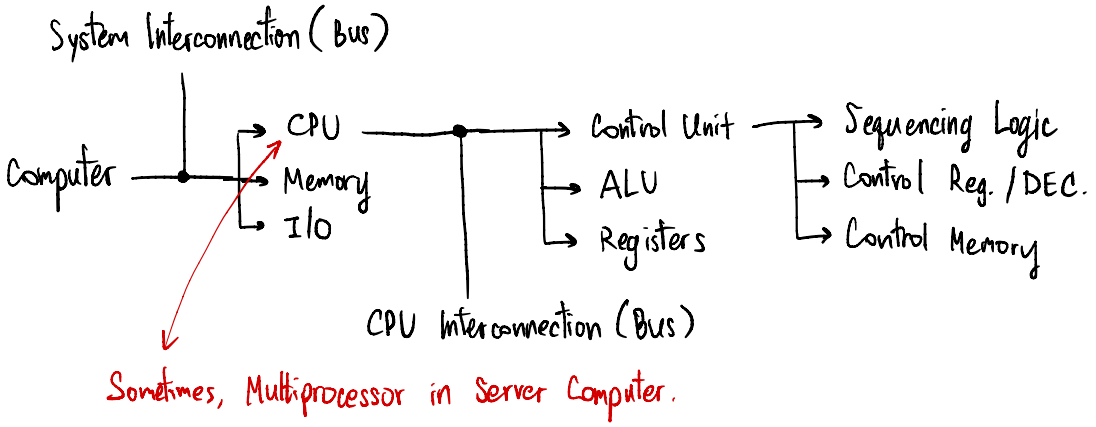
EMBEDDED SYSTEM

Introduction

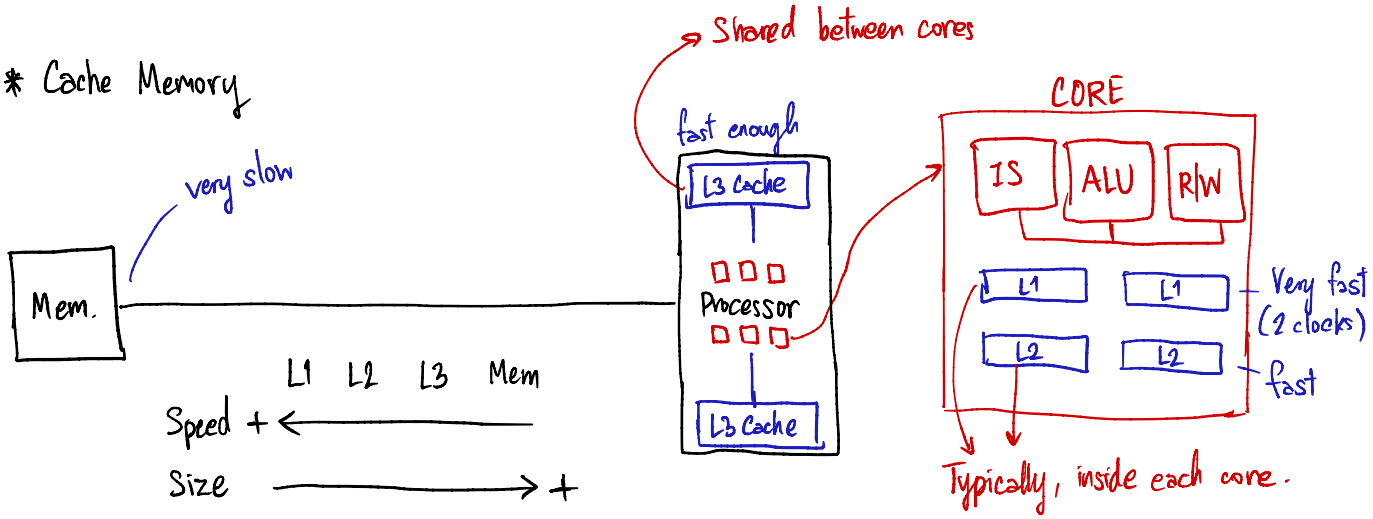
* Instruction Set Architecture (ISA) - Instruction Set
  - I/O
  - Mem. Addressing, endianness
* Computer Organization - Specifications
  - Inter-Computer Comm.

Brief History : Vacuum Tube → Transistor → Integrations (IC)
→ IBM 370 Arch., ...

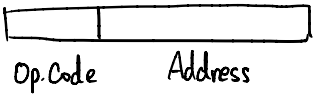
Designing → Hierachical System → Structure
→ Function - Data Processing
- Data Storage
  - Short-term
  - Long-term
- Data Movement
  - I/O
  - Communications, Protocols
- Control



* Cache Memory



Instructions (Basic)



→ "The Beginning" : Intel 8086

Architectures (ISA) ┬ CISC (Complex) (Many ops. (Micro operations) happen per instruction (operation)) → Intel x86
└ RISC (Reduced) (1 Operation (or few) per instruction) → ARM

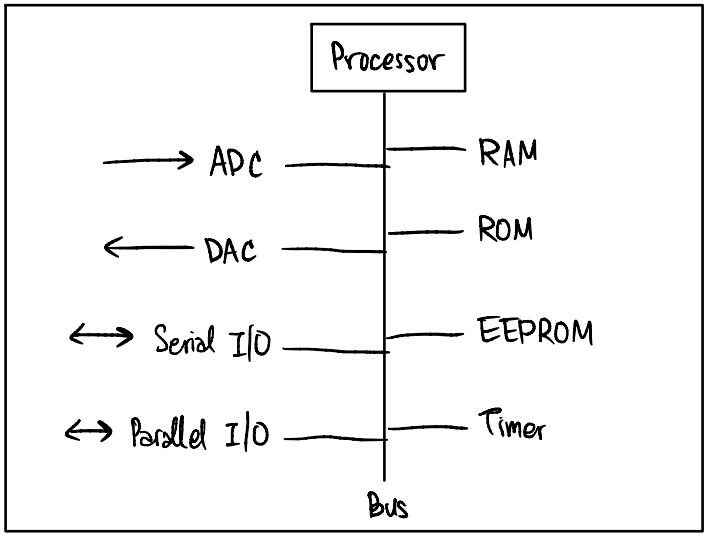
Embedded System : Internal system in a larger system or a standalone system.

* Real-time constraints

Internet of Things : ES with Internet

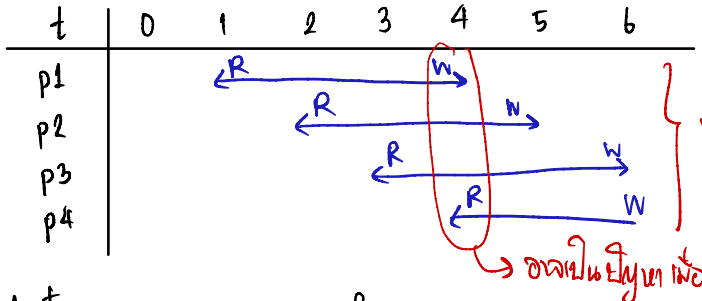
- Information Technology
- Operational Technology
- Personal
- Sensor / Actuator

Microcontroller

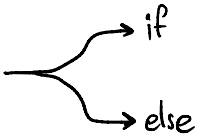


Performance Enhancement

1. Pipelining



2. Branch Prediction :



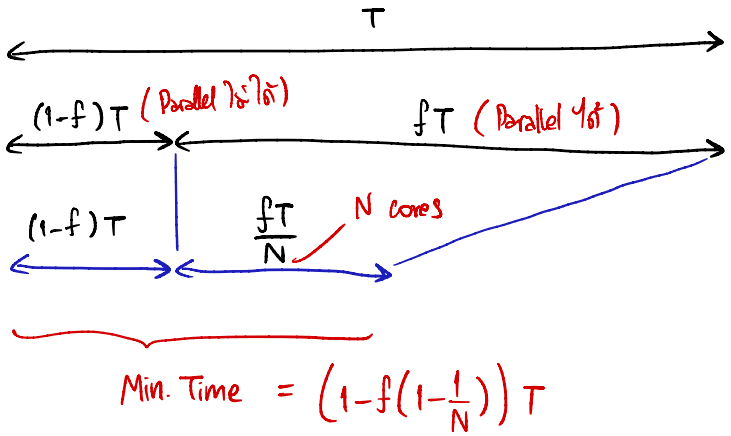
3. Superscalar Execution : Many instructions per clock cycle

4. Data Flow Analysis : Data depends on result หรือไม่? ต้องรอผลลัพธ์ก่อนหรือไม่?

5. Speculative Execution : 2. + 4. ทำคาดล่วงหน้าเมื่อ execution engine ว่าง

Amdahl's Law

Assume workload T



Min. Time = $\left(1-f\left(1-\frac{1}{N}\right)\right)T$

Little's Law

: In Queueing System, $\langle N \rangle = \langle \dot{N} \rangle \times t$

N : Number of items

$\dot{N}$: Rate which items arrive.

↳ ยังใช้ได้กับ multiple queues, ...

Clock :

หน่วยการชี้ความเร็ว : MIPS (million instructions per second)

FLOPS (floating-point logic operations per second)

↳ MFLOPS, GFLOPS, TFLOPS, ...

OPS (operations per second)

Benchmarking : การหาค่ากลาง สำคัญที่ G.M., ทำ p-test

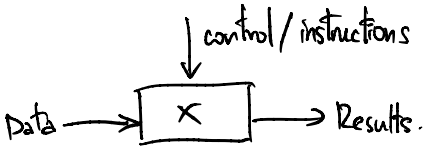
- Easily measured
- High-level, portable
- Wide-distribution
- Represents prog. lang. paradigms

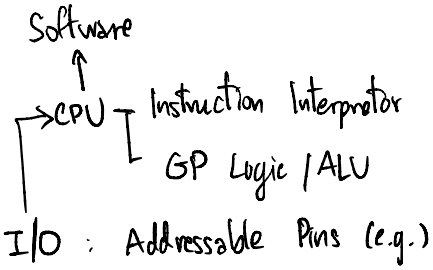
e.g. SPEC Benchmark

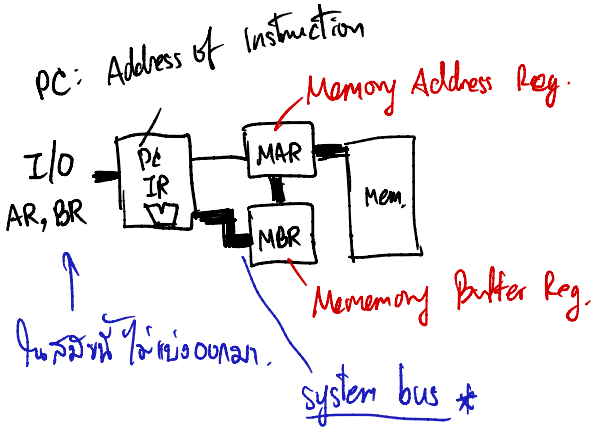
Computer Interconnections & Functions

*Neumann Architecture : sequential instruction.

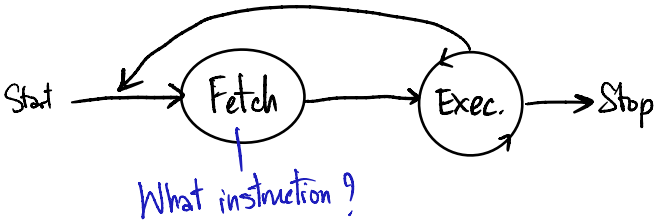
- Single R/W mem.
- Addressable mem.







Fetch-Execute Cycle



1. Pull instruction from mem.
2. PC increments.
3. Load instruction into IR. (Instruction Register)

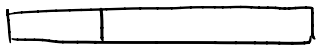
Actions

1. Processor ↔ Memory (Cacheable)
2. Processor ↔ I/O (Not cacheable)

} Data Transfer

3. Control : Instruction specifying sequence of exec.
4. Data Processing : Arithmetic Operation, Logic Operation, ...

Instruction Format



Op.Code — what to do.
Address — where to do.

Instruction Fetch & Exec. อยู่ใน cycle เดียวกันไม่ได้ (ถ้าพร้อมกัน) : Mem. R/W ได้แค่: 1.

i.e., ใน RISC Arch. : ทำแค่นี้.

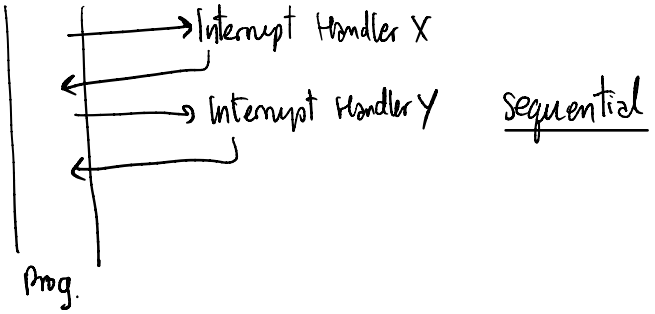
Levels of Interrupt: Program e.g. 0#DIV, Ref. outside alloc. mem., ...

Timer : Do something now. / Switch between processes.

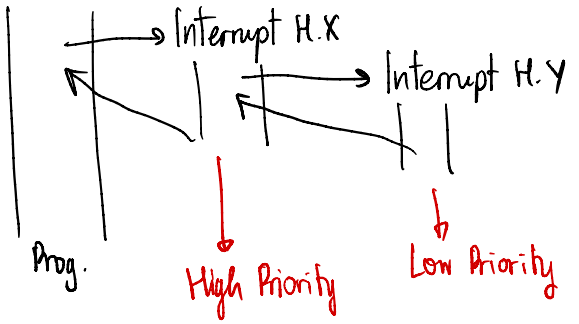
I/O : Signal completions/errors to CPU by I/O controller.

Hardware Failure : Power failure / Parity error in Mem.

Interrupt Processing :



sequential

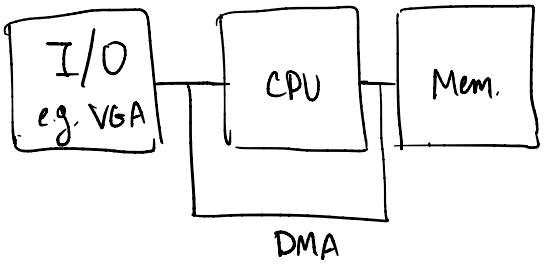


nested / NVIC

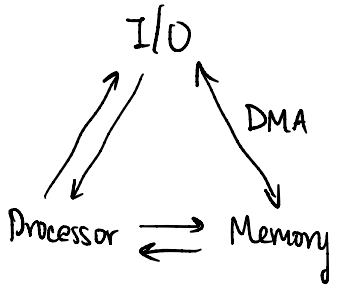
I/O Functions

* I/O communicates directly with processor.

Want : I/O comm. with mem. directly. : DMA (Direct Memory Access)



;

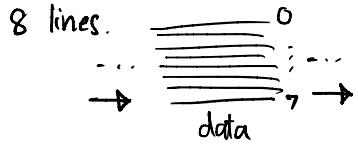


(Ideally, but not practical.
Introducing "Bus".)

Bus Data Bus : Moving data between modules with line (parallel according to bit width)

e.g. 128 lines, 0-63, 64-127

Problem : Line length



Address Bus : Signalling locations.

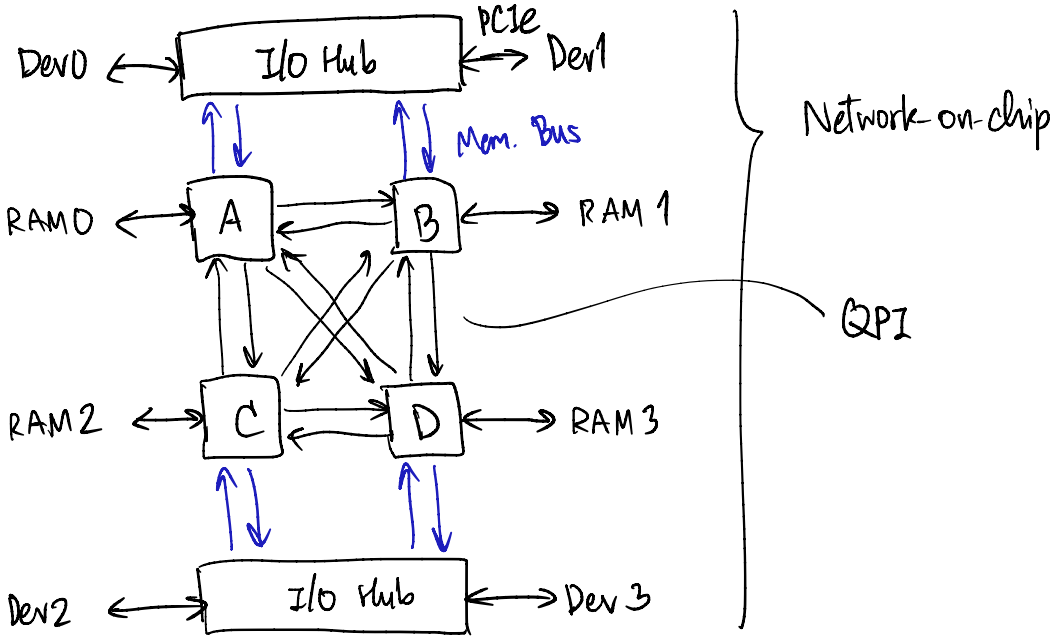
control Bus : Signalling functions.

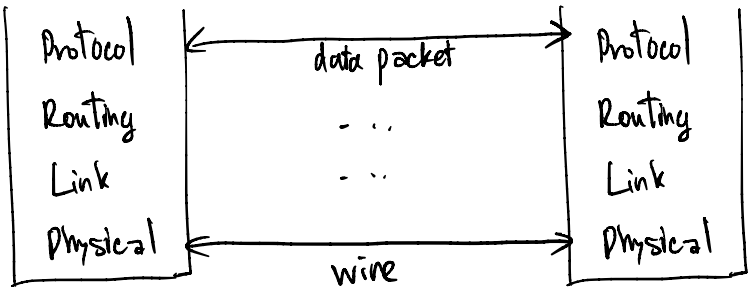
Point-to-Point Interconnection

QPI (2008) : Pairwise connections (P2P)

Layered protocol (e.g. OSI Network Layer)

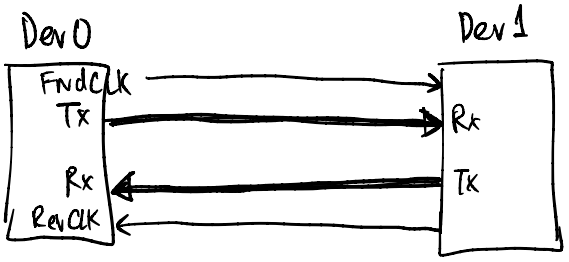
Packet of data : header – data – ECC.





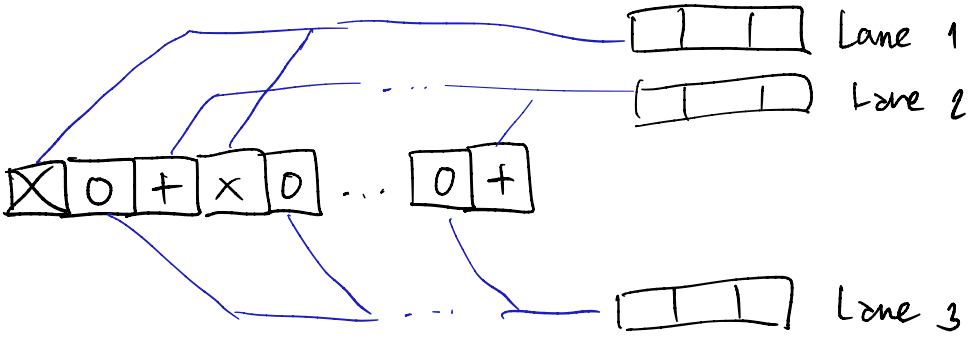
QPI layers

Lane : Group of data path / set of wires.



1 lane

Multilane

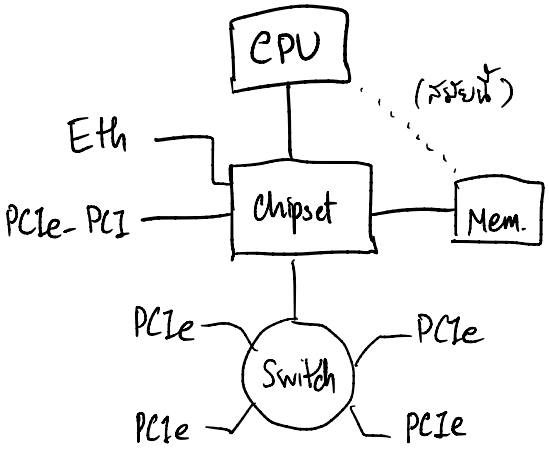


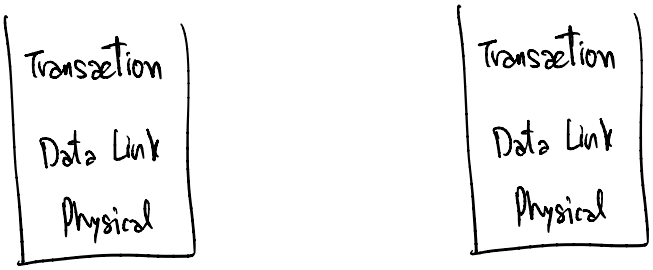
PCI : Peripheral Component Interconnection

↳ Separate Bus , เป็น Bus-based schematics

PCIe (PCI Express) เป็น P2P มี PCIe Controller / Switch

∴ เป็น Multilane.





PCIe Layer

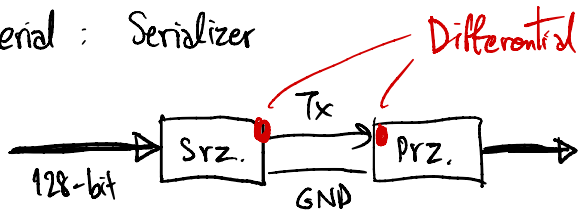
* Transaction คือ set ของ operation ที่ต้อง successful ทั้งหมด. ถ้าไม่ ให้ Rollback ทั้งหมด.

e.g. ฝากเงิน → ปล่อยสินเชื่อ → จบ

คืนเงิน ← if failed

(Rollback operation)

Parallel → Serial : Serializer



PCIe TL (Transaction Layers) – Request, Mem. Access, Config, I/O, ...

* PCIe บางแบบ สามารถทำ DMA ได้ e.g. GPU, Accelerator, TPU, ...
  - ที่ชั้น TL (Layer : Encapsulate higher layer onto lower layer).
- Implementation บน PCIe เช่น NVMe

Communication Protocols

e.g. EDID in HDMI

- Asynchronous : No clock at all.
- Synchronous : with clock from Master

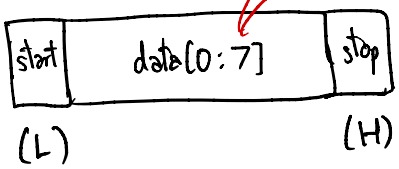
- Serial : Data stream
- Parallel : Data width

Modes
- Simplex →
- Half Duplex ↔
- Full Duplex ⇄

UART (Universal Asynchronous Receiver/Transmitter)

- Protocol , ไม่ใช่ Hardware Implementation
- Standards ที่ใช้ UART: RS-232, RS-485, ... ต่างกัน Voltage, Comparison, ... ของ hardware.

Packet



use low #bits : clocks difference/deviation → offset.
can be any bits (e.g. 5 bit, 8 bit, 7 bit ...)

Bit timing : Baudrate (bps)
(switching time

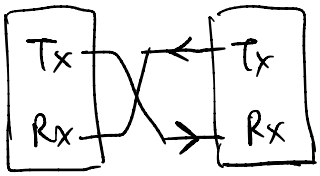
standards.

"9600-N-8-1" or 9600-8-N-1

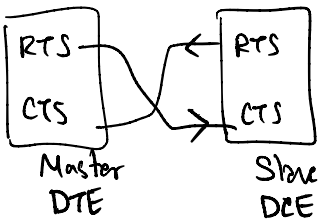
- stop bit
- data bits
- parity
- baudrate

* Standards ต้องตรงกันทั้ง 2 ฝั่งก่อนเริ่ม *

- RXD (รับจาก TXD)
- TXD (ส่งให้ RXD)



- RTS # Ready to send
- CTS # Clear to send



DTE ส่ง RTS ให้ DCE
เพื่อขอส่งข้อมูล จาก DTE → DCE
DCE ตอบกลับด้วย CTS ให้ DTE

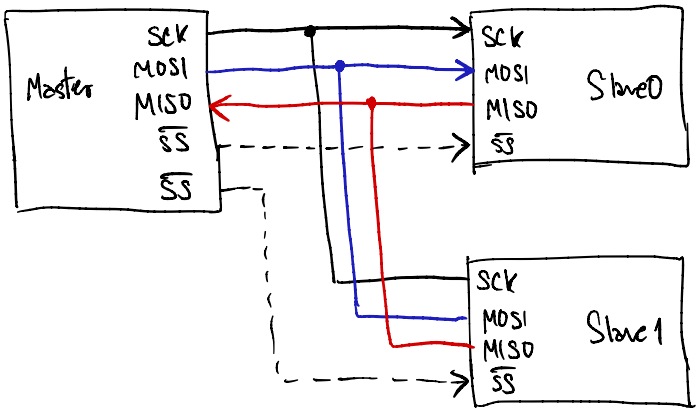
±12 V Lower Interference, High Signal Integrity.

UART $\xrightarrow{\phi}$ RS-232 ϕ: Inverse mapping: $[0, +5] \rightarrow [+12, -12]$

SPI (Serial Peripheral Interface) — ใช้ความเร็วสูงๆ , clock สูงๆ / MHz.

└ Synchronous, Full-Duplex, Master-Slave

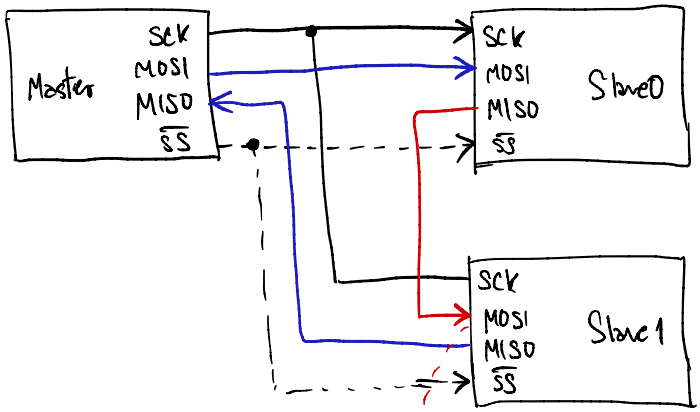
Multiple Slaves



Daisy chain Multiple Slaves

* Daisy chaining

สัญญาณจากอุปกรณ์ที่ส่งออกไป.



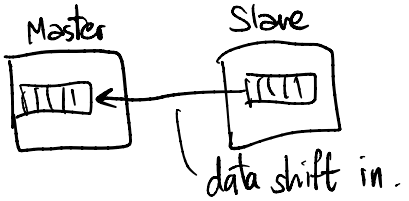
Communication (Hardware)

MOSI ⟶ } Data Path

MISO ⟵ }

SCK ⟶ } Control Path

SS ⟶ }

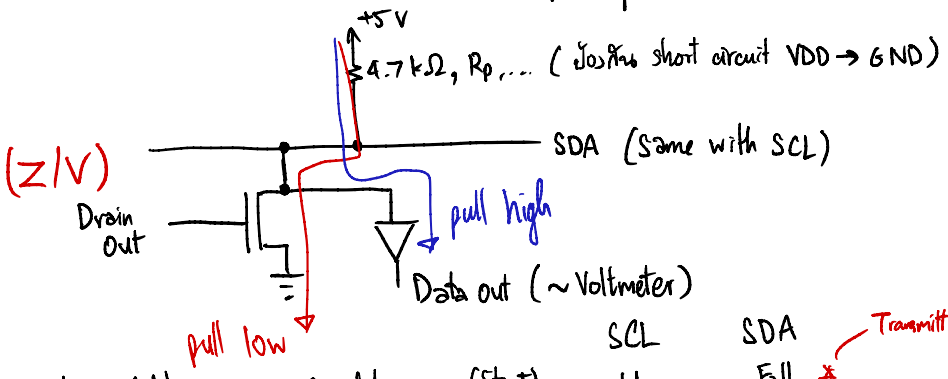


Data Validity : Design ให้ Data change เมื่อ Falling

Data Latch (Read) เมื่อ Rising } CLK

I^2C (Inter IC / Inter Integrated Circuit)

- Synchronous, Half-Duplex, Multi-Master-slave.
  - O SDA : Data Line
  - O SCL : Clock Line
  - Software Addressable
  - Multi-master/slave with CD (collision detection)
  - Serial, 8-bit, Bi-directional, 4-mode
  - On-chip filtering
- Wired-AND (Open-Drain) : ต้อง pull-up resistor.



Data Validation : Condition

	SCL	SDA
(Start)	H	Fall *
(Stop)	H	Rise

* Transmitter ส่ง data ลง

* Data changes on clock L. , latches on clock H.

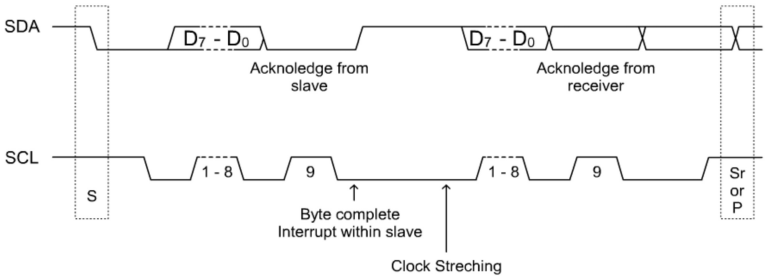
(*) Wire-and Signal ใช้ pull-up ต้องใช้ resistor เพื่อ signal ตัวอื่นได้ (*)

I2C ส่ง MSB first LSB last (D7D6D5...D1D0)

(D7 D6 ... D1 D0)

8 bits

- Acknowledge on bit 9 (with SDL Low)
- If ack is slow, clock from master can "stretch".



From Slide

Clock Synchronization (Multi-master)

- One clock will wait for slower clock.

Data Arbitration : If 2 devices crash from calling same address, devices always check for data, if it is wrong, that device will stop transmitting.

Data payload : (Frame)

- 1–7 Addr.
- 8 R/W (H/L)
- 9 \Ack (NACK)
- 1–7 Data
- 8 Data
- 9 \Ack.

* NACK ที่ Addr, RW = no slave
* NACK ที่ Data = slave รับไม่ได้ (W) NO ACCEPT
* NACK ที่ Data = ไม่ slave ส่งต่อ (R)

I^2C Speed Standard

Wire-AND

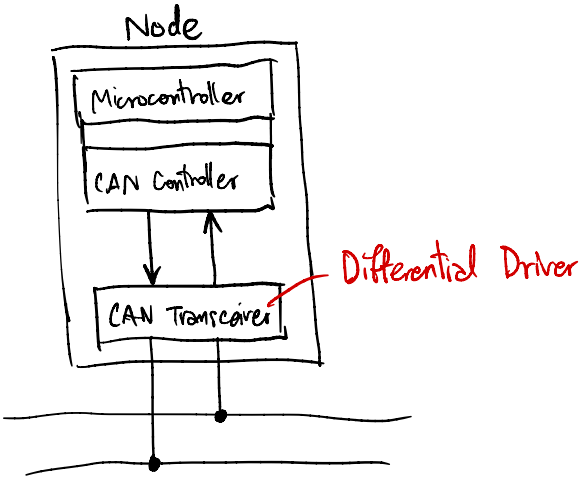
Standard	100	kbps
Fast	400	kbps
Fast+	1	Mbps
High Speed	3.4	Mbps

} Bidirectional

Ultra fast mode 5 Mbps (Unidirectional), uses push-pull driver.

CAN (Controller Area Network)

- Asynchronous, half duplex, Multi master/slave.



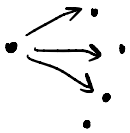
- Error Counters

Identifier/Address

11-bit ⟶ 29-bit

Concepts

1. Messages Prioritization
2. Latency Guarantee (min. time)
3. Multicast with time sync.



(ไม่เหมือน broadcast)
เหมือน many unicasts

4. Error detection, signaling
5. Auto Re-transmission.

USB (Universal Serial Bus)

- Asynchronous, half-duplex?, master-slave.
- Power management

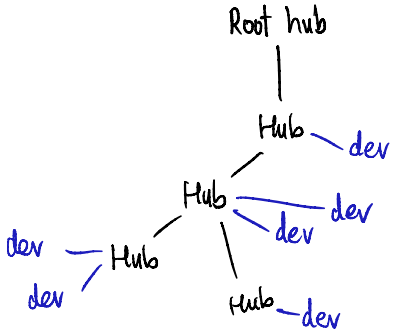
* Upstream, Downstream connection

1.x – 2.0 : Vcc, D+, D-, GND

3.0 : SSTX+/-, SSRX+/-

Devices:
1. Host
2. Hub / Root Hub
3. Peripherals

Topology



* ต้องมี Enumeration ทุก 1 อุปกรณ์ เพื่อ identify อุปกรณ์นั้นๆ (แล้วให้ address)

Hub
1. Bus-powered
2. Self-powered

Peripherals
1. Standalone (1 addr.)
2. Compound (many addr. in 1 dev.)

(USB)

Power Management

* Host must be able to disable power from 1 device.
  └ แล้วไม่ได้ implement, ไม่ได้นอก.

Endpoints : 16 endpoints
each 4-bit addr.

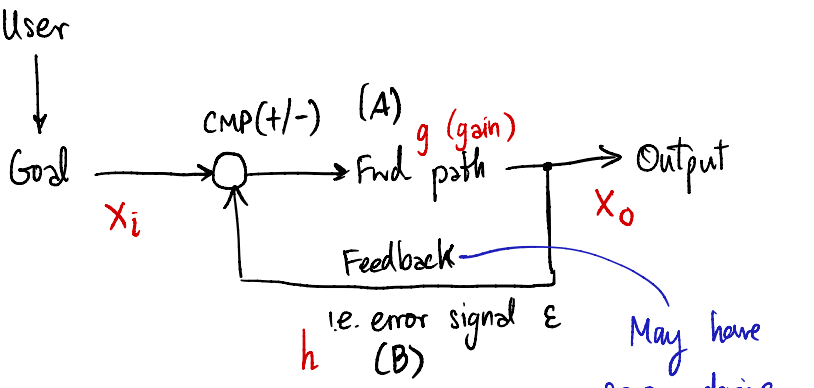
Transfer types
1. Control : Config. & setup
2. Bulk : Sends data over time. e.g. files, ...
3. Isochronous : Mouse / keyboard, data streaming
4. Interrupt : Polling

Controls & Feedback

- Regulation or Command
- Goal (Target value)

- Open-loop control : User → Goal
  - User tries to achieve goal
  - Fwd. path → Output

- Closed-loop control :



Linear control

$$X_i - BX_o = \frac{X_i}{A} \longrightarrow G = \frac{X_o}{X_i} = \frac{A}{1+AB}$$

A — open-loop gain

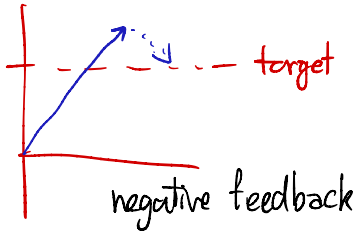
G — overall gain "closed-loop gain"

(G>A) AB neg. → positive feedback (e.g. set ค่าผิด, วงจรสั่น) → unstable

(G<A) AB pos. → negative feedback

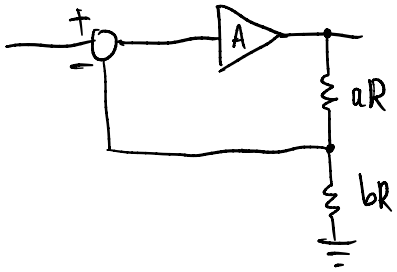
↳ if $AB \gg 1$: $G \approx \frac{A}{AB} = \frac{1}{B}$ ไม่ขึ้นกับ Fwd. path

B — feedback only

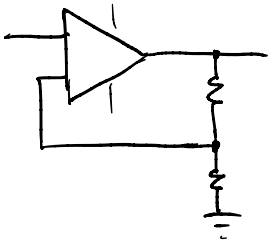


G · Bandwidth = const.

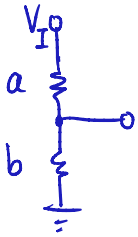
Intro to Operational Amplifier



OpAmp.



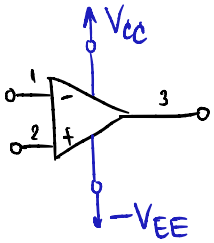
Voltage Divider (V. Div)



$\frac{V_I}{a+b} = \frac{V_O}{b}$

$\therefore V_O = \left(\frac{b}{a+b}\right)V_I$

Operational Amplifier



1 : Inverting input

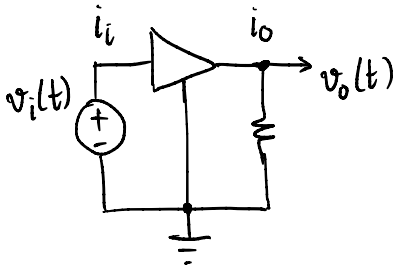
2 : Non-Inverting input

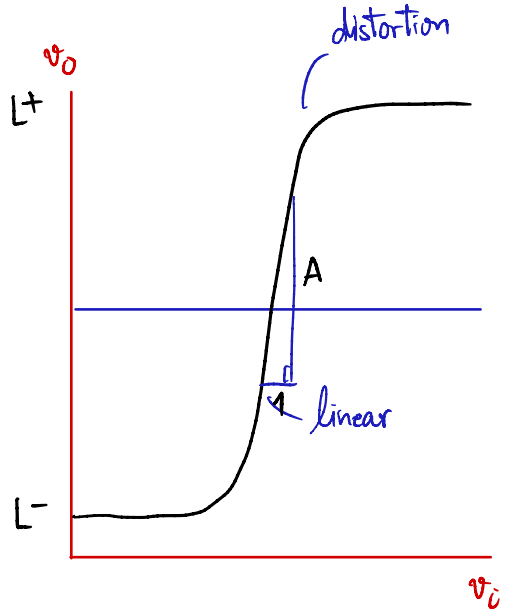
3 : Output

เป็น Active/IC

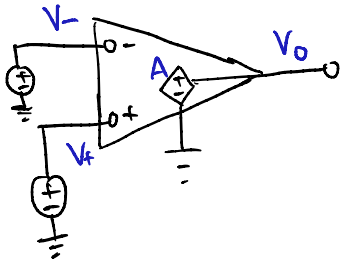
ต้องใช้ไฟเลี้ยง (V_{CC})

Simple Amplifier





Ideal OpAmp

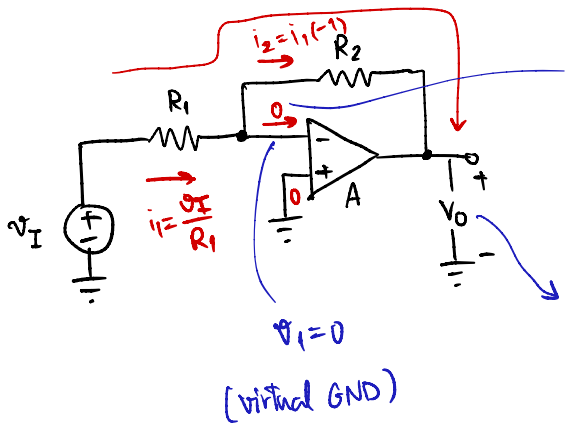


1. Infinite Input Impedance (Disconnected)
2. Zero output Impedance
3. Zero common-mode gain
4. Infinite open-loop gain (A)
5. Infinite bandwidth

$V_O = A(V_+ - V_-)$

$V_O \propto \Delta V_{in}$, ไม่สนใจ $|V|$

- Inverting closed-loop configuration (Ideal OpAmp)



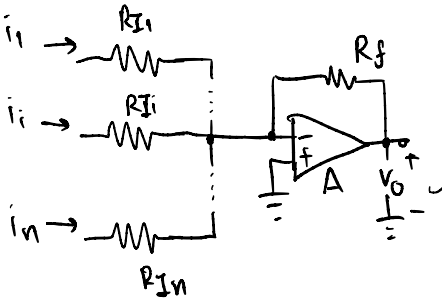
∵ Infinite Impedance → $I = 0$.

$\left(\frac{V_O}{A} = V_+ - V_- \rightarrow V_+ = V_-\right)$, $A \rightarrow \infty$

$$\therefore V_O = 0 - i_1 R_2 = -\frac{V_I}{R_1} R_2 = -\frac{R_2}{R_1} V_I$$

If $R_1 = R_2$, "inverter" $V_+ \rightarrow V_-$

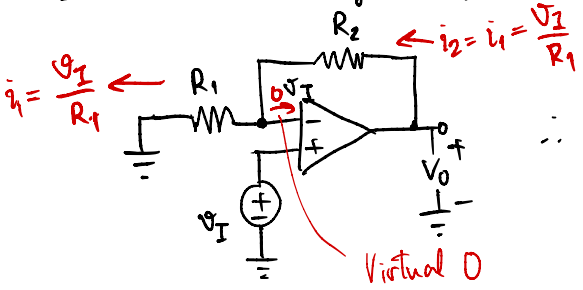
- Weighed summer



$$\therefore V_O = -R_f\left(\frac{V_1}{R_1} + \frac{V_2}{R_2} + \ldots + \frac{V_n}{R_n}\right)$$

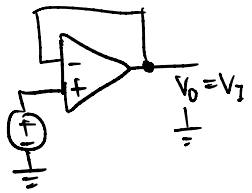
- Inverter 2 ตัวต่อกัน → non-invert

<u>แบบ</u> • Non-inverting config.



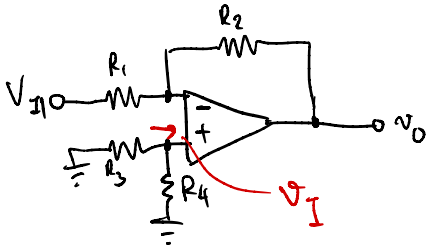
$$\therefore V_O = V_I\left(1 + \frac{R_2}{R_1}\right) \quad : \text{Gain} \geq 1 \text{ เสมอ}$$

- Voltage lowerer (Unity-gain buffer) / Follower

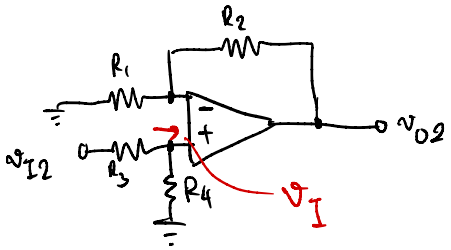


→ Signal Repeater

• Difference Amplifier



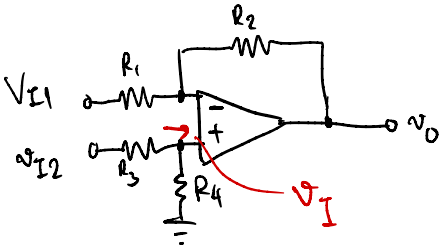
$$v_{o1} = -\frac{R_2}{R_1} V_{I1}$$



$$v_{o2} = \left(1 + \frac{R_2}{R_1}\right) v_I \quad \text{(V-div.)}$$

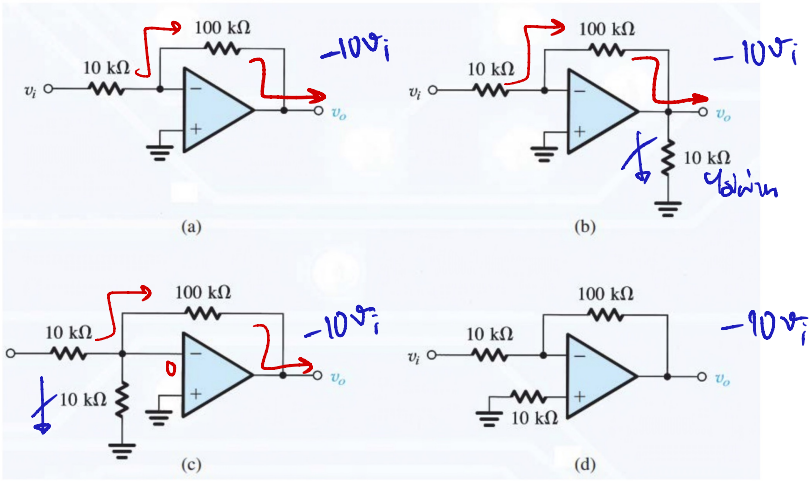
$$= \left(1 + \frac{R_2}{R_1}\right)\left(\frac{R_4}{R_3 + R_4}\right) v_{I2} \Rightarrow \frac{R_2}{R_1} V_{I2}$$

$$\frac{R_1}{R_2} = \frac{R_4}{R_3}$$

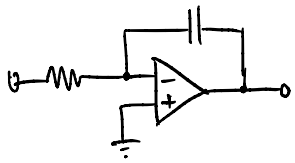


$$\therefore V_o = V_{o1} + V_{o2} = \frac{R_2}{R_1}(V_{i1} - V_{i2})$$

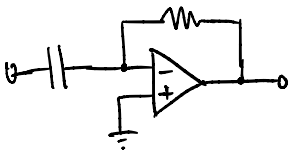
e.g.



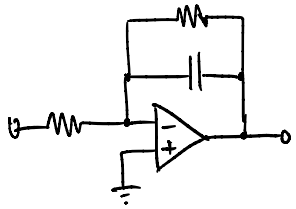
With C :



(Integrator)



(Differentiator)

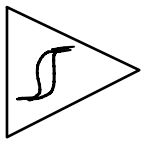


(Filter)

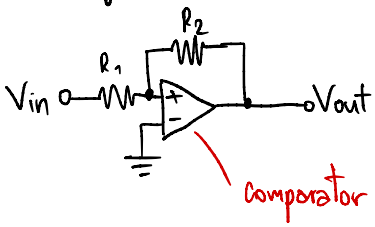
Comparators

↓

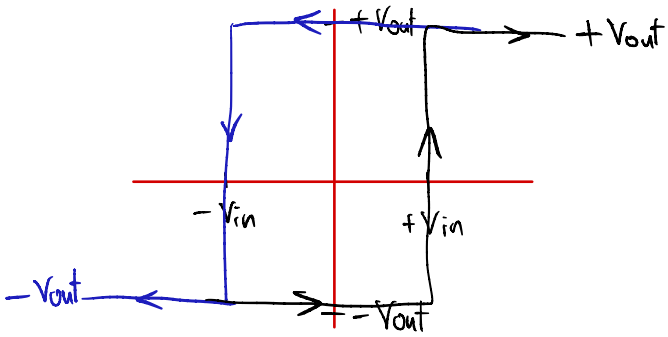
Schmitt Trigger



(Inverting)



: ใช้กัดจัด Noise โดยการตั้ง Threshold

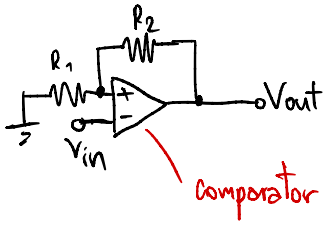


e.g. $R_1 = 1k\Omega$

$R_2 = 2k\Omega$

$V_0 = -\frac{1}{2}V_{in}$

(Non-Inverting)

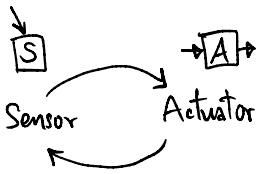


$V_0 = -\frac{R_1}{R_1+R_2}V_{in}$

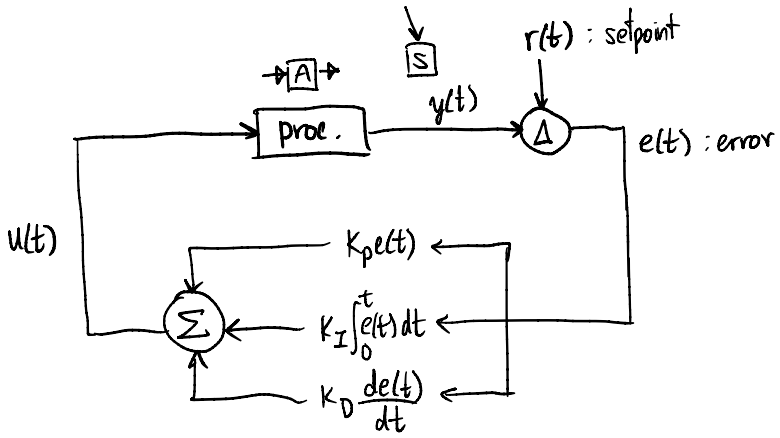
Homework

LTspice

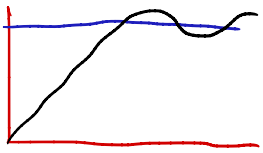
PID Control

* Feedback Loop : 

e.g. Ⓜ

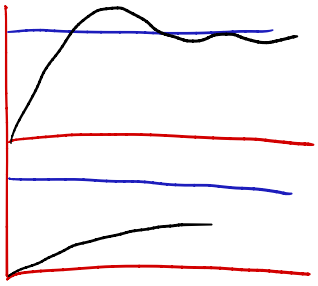


- Simple
  - PWM On if lower, Off if higher than setpoint. → oscillate

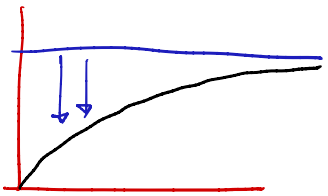


- Proportional (P) : Power
  - output = $K_p \cdot error$.

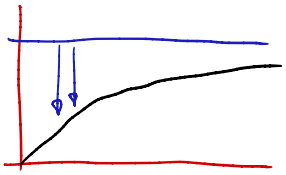
  Problem : Too much → oscillate
  Too low → never reach setpoint.



- Proportional-Integral (PI) : Position
  - output = $K_p \cdot error + K_I \cdot accumulated\_error \cdot \Delta t$



- Proportional-Derivative (PD) : ต้องรู้ speed รู้เลยว่าไกล, จะเพื่องใกล้ : Speed
  - output = $K_p \cdot error + K_D \cdot \frac{(e(t) - e(t_{last}))}{\Delta t}$



→ PID : Good position + Good speed

* Saturation (clamp output!) : MIN_OP ≤ output ≤ MAX_OP.

* Timesteps : เก็บตัวแปรเวลา timestep ออกมา K_I, K_D

Tuning P : 0 → should be at full power
I : 0 ——— → until steadystate is OK.
D : 0 ——— → dampen overshoot & improve responsiveness

Application / Uses : "Cascaded PID"

* Using OP Amp. to make PID ต่อ

Booting Process

Know: Interrupt Vector : ตัวเก็บ addr. อันนึง

Interrupt - DMA

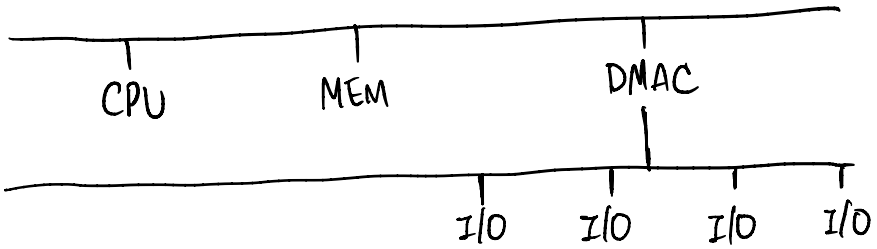
I/O - Programmed : Polling
- Interrupt-driven

} need CPU intervention

* Interrupt มาจาก I/O
* Exception มาจาก CPU

- Direct Memory Access → uses DMA controller

DMA Controller Config.



ARM Interrupts

- Non-Maskable Interrupt (NMI) : Ignore ไม่ได้!
- Interrupt Request (IRQ) / Masking

Tail-chaining : Push next interrupts onto the stack.
Then pop all at once once done.

IVT (vector table) : Lookup table for Types, priority, loc. of handler