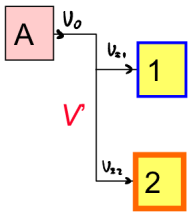
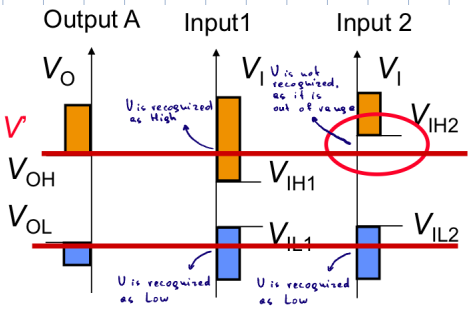
B → Digital Electronics





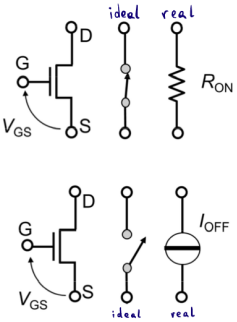
$V_{OH} > V_{IH}$
$V_{OL} < V_I$
} condition for V to be recognized

where:

- V_{OH}, V_{OL} are the voltages inputted
- V_{IL}, V_{IH} are the voltages to be compared to

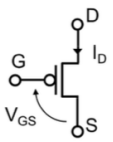
NMOS as a switch

- Two states
  - ***Switch closed: we apply $V_{GS} >> V_{th}$***
    - » ON state (closed) → I_D flowing
    - » ideal: short circuit
    - » real: R_{ON} resistance
  - ***Switch open: we apply $V_{GS} << V_{th}$***
    - » OFF state (open) → no I_D
    - » ideal: open circuit
    - » real: I_{OFF} leakage current (sign unknown, linked to V_{DS})

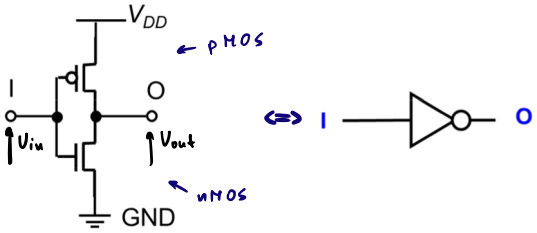


PMOS as a switch

- Voltages and currents have opposite sign with respect to the nMOS
  - $V_{GS} < V_{TH}$: ON state (conducting)
  - $V_{GS} > V_{TH}$: OFF state (non conducting)
- Complementary controlled switch

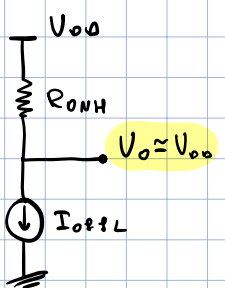


Inverter realization

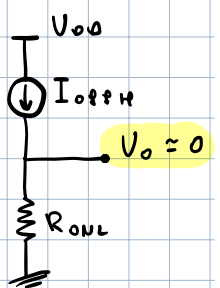


CMOS structure

$V_{in} < V_{th}$
(usually $V_{in} = 0$)

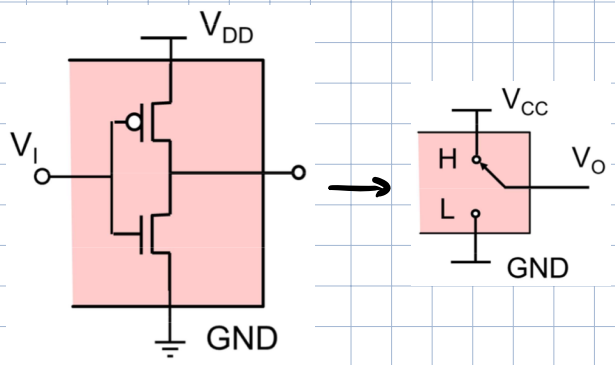


$V_{in} > V_{th}$
(usually $V_{in} = V_{DD}$)



equivalent circuit

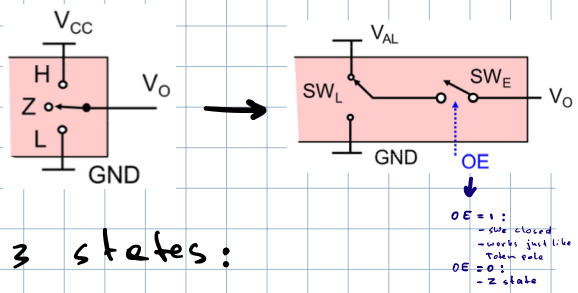
Totem pole



2 states:

- H → $V_o \simeq V_{cc}$
- L → $V_o \simeq 0$

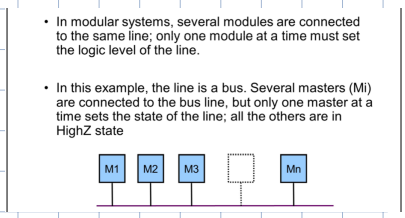
3 states output circuit



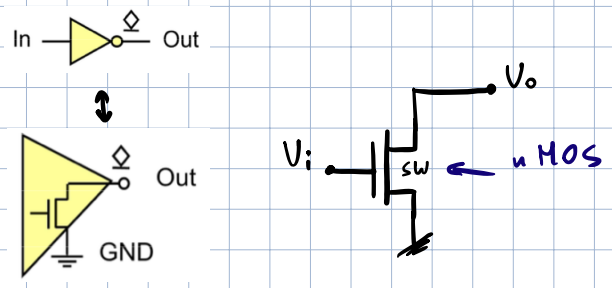
3 states:

- H → High
- L → Low
- Z → High Impedance
  - ↳ V_o is O.C.
  - ↳ mostly used to disconnect the output voltage

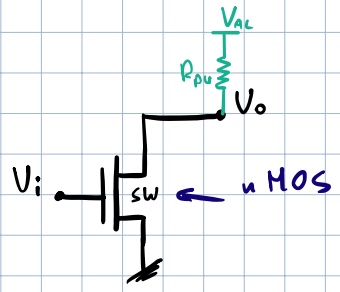
why do we need 3s output?



Open Drain output

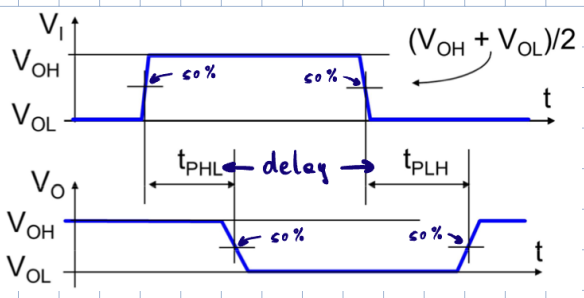


- realized with a single NMOS
- sw open → Z state
- sw closed → $V_o \simeq 0$



- if we add a Resistor at the output:
  - sw open → $V_o \simeq V_{AL}$

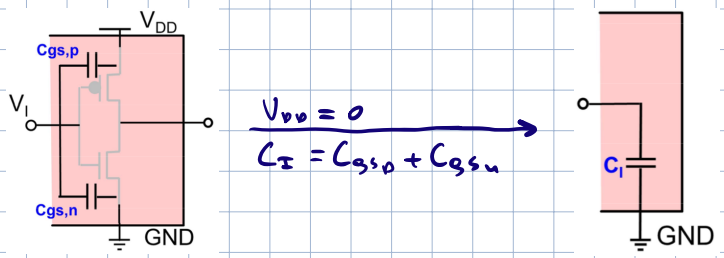
Delay btw input/output signals



input

output

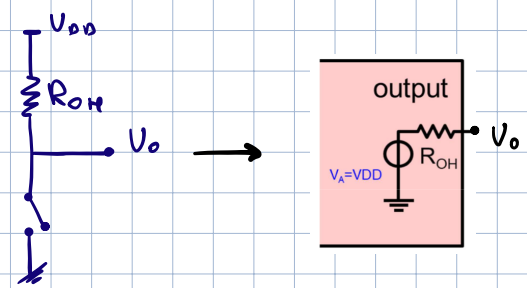
Where does the delay come from ?



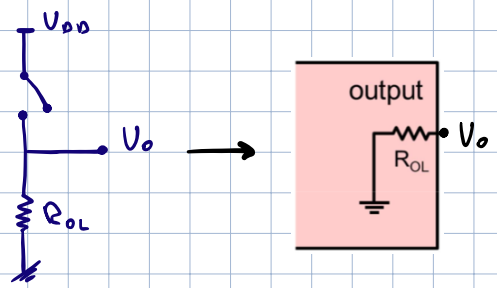
C_{gsp}, C_{gsn} are the capacitances btw the source and the ground of the MOSFETs

input equivalent circuit

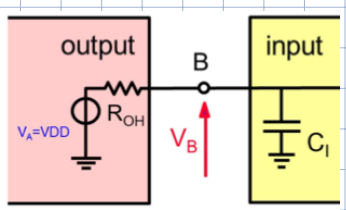
H state:

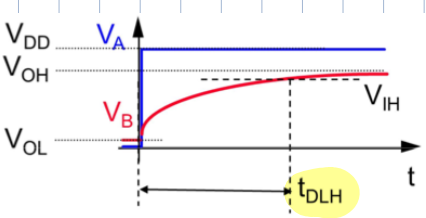


L state:



L → H

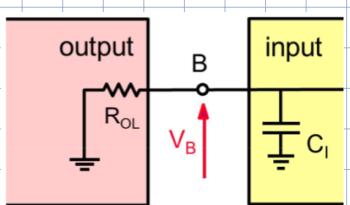


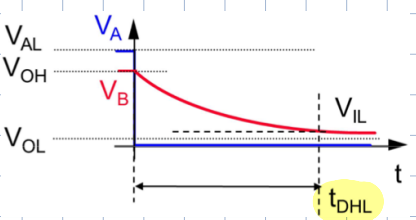


during the transmission Ci is charged → delay

$t_{DLH} \approx 0.69\ C_i\ R_{OH}$

H → L

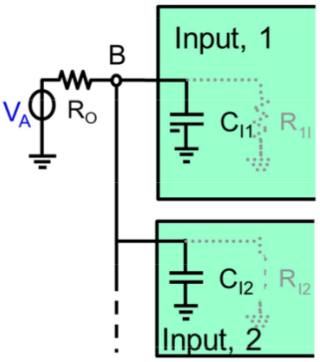




during the transmission C_i is discharged → delay

$t_{DHL} \approx 0.69\ C_i\ R_{OL}$

Multiple inputs



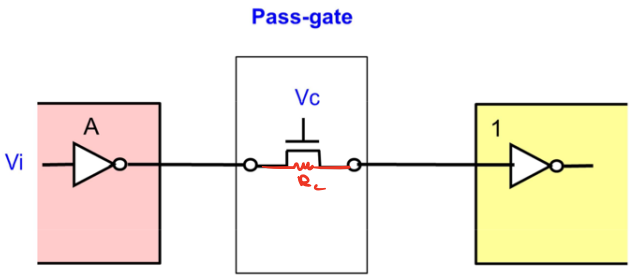
$t_{OLH} = C_T \cdot R_O$,

where $C_T = sum(C_i)$

The maximum number of inputs that can be connected to a single output is limited by the maximum delay time set by the application.

↳ Fan Out

Delay with Pass gate

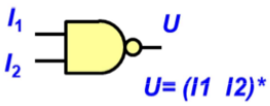


$t_D = 0{,}69\,(R_C + R_{ON}) \cdot C_i$

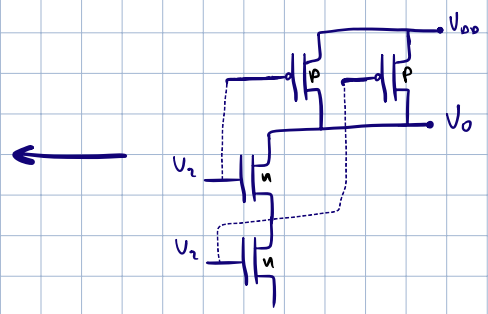
it adds additional delay

Logic gates Realizations with CMOS

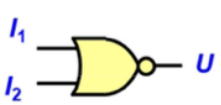
NAND



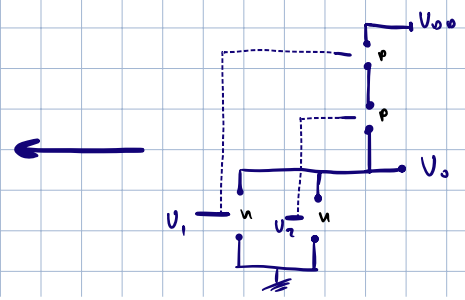
I_1	I_2	U
0	0	1
0	1	1
1	0	1
1	1	0



NOR



I_1	I_2	U
0	0	1
0	1	0
1	0	0
1	1	0



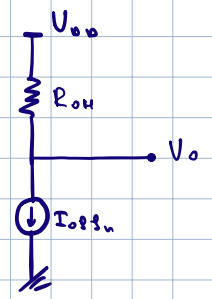
- To realize any combinational logic function, we can use a combination of series/parallel nMOS and pMOS switches
  - » If nSWs are in series → pSWs are in parallel
  - » If nSW are in parallel → pSW are in series

Power dissipation

Static ← DC, after the transmission has completed

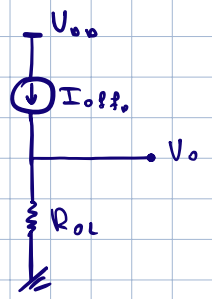
C_i is full

H state



$P_s = V_{DD} \cdot I_{off_n}$

L state



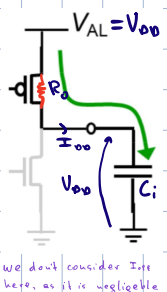
$P_s = V_{DD} \cdot I_{off_p}$

⊗ We consider I_{off} here, as many numbers of transistors actually make up a significant static power dissipation

Dynamic ← AC, during the transmission

C_i is being charged/discharged

L → H



⊗ We don't consider I_{off} here, as it is negligible

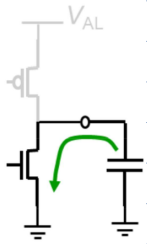
$$\underbrace{f\,Q}_{I_{DD}} = C_i \cdot V_{DD}\, f$$

$$P_D = V_{DD} \cdot I_{DD}$$

$$P_D = V_{DD}^2 \cdot C_i \cdot f$$

where f is frequency

H → L



$V_o = 0 \rightarrow P_D = 0$

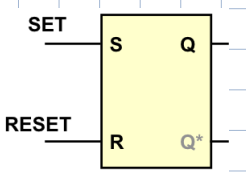
- L → H dynamic Power can be <u>reduced</u> by:
  - reducing the frequency f (upper limitation) $\rightarrow f_{max} = 1/t_p$ (t_p: delay)
  - reducing V_{DD}
  - reducing C_i

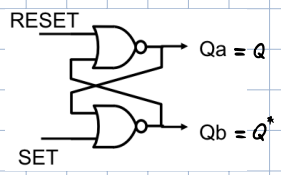
Sequential Circuits

Combinational logic circuits:
the output at time instant t_0 depends **ONLY** on the inputs at the same time instant $t = t_0$

Sequential logic circuits:
the output at time instant t_0 depends on the inputs at $t=t_0$ **AND ALSO** by the previous states at $t = t_{0-1}, t_{0-2}, \ldots$)

Latch Set Reset



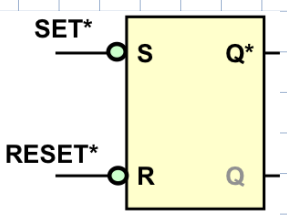


S	R	Q	Q*
0	1	0	1
1	0	1	0
0	0	Q_{-1}	Q^*_{-1}
~~1~~	~~1~~	~~0~~	~~0~~

← memory

metastability state

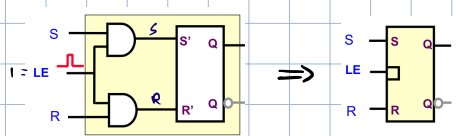
Latch-SR with NAND



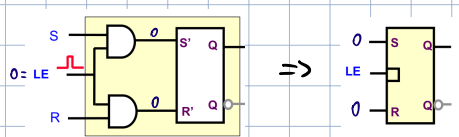
S*	R*	Q	Q*
1	0	0	1
0	1	1	0
1	1	Q_{-1}	Q^*_{-1}
~~0~~	~~0~~	~~1~~	~~1~~

← memory

Gated Latch-SR (with enable signal)

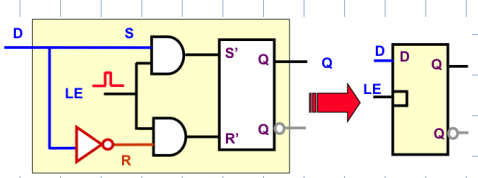


← LE = 1



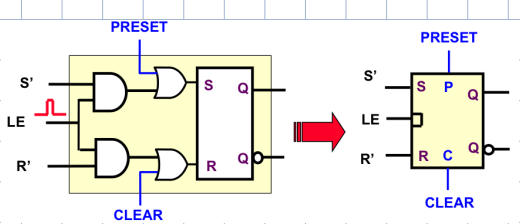
← LE = 0

when LE = 0, Q does not change even if S or/and R changes



← Gated D-Latch

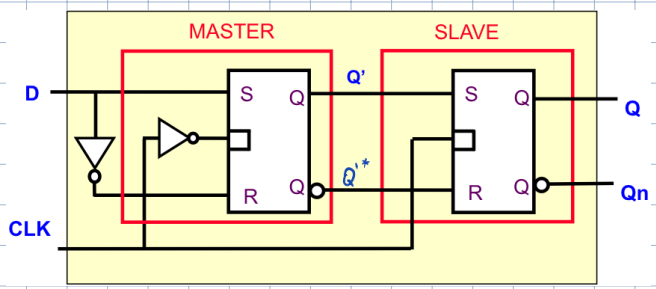
Preset / Clear command to bypass Enable



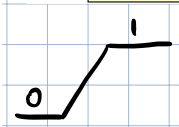
Preset = 1 → Q = 1

Clear = 1 → Q = 0

D-Flip Flop



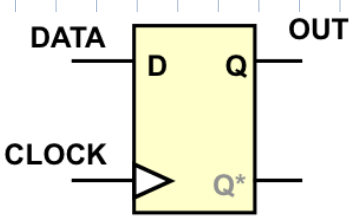
CLK	D	Q'	Q
0	0	→ 0	Q_{-1}
0	1	→ 1	Q_{-1}
1	0	$Q'_{-1} = D_{-1}$ →	D_{-1}
1	1	$Q'_{-1} = D_{-1}$ →	D_{-1}



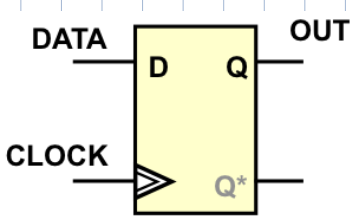
→ during the transition from 0 to 1, the output stay with the last value of D, so even if D changes while CLK = 1, Q is always equal to the last value of D before the transition.

for CLK on the rising edge

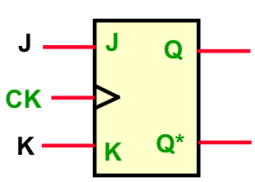
on rising edge



on both edges



JK Flip Flop



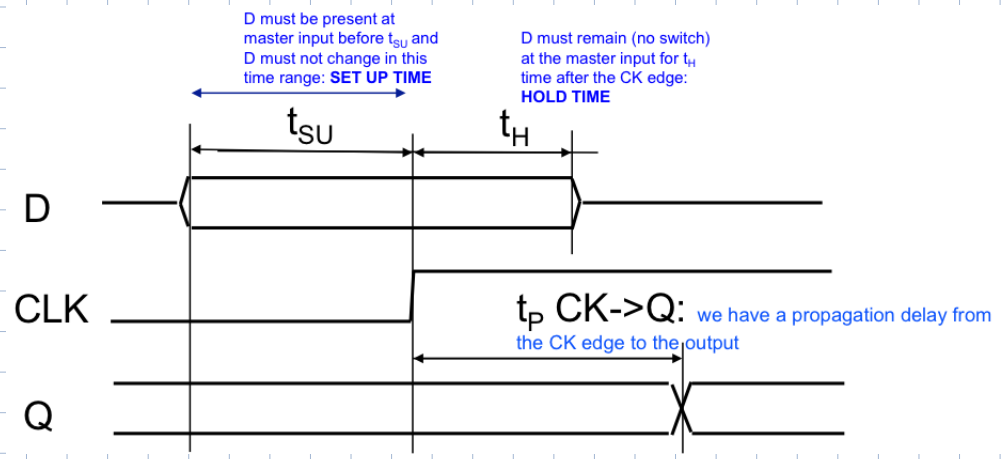
J	K	Q	Q*
0	0	Q_{-1}	Q^*_{-1}
0	1	0	1
1	0	1	0
1	1	Q_{-1}*	Q_{-1}

Memory state

per J=K = 1: the output switches at each CK edge (*toggle*)

J or K set the output

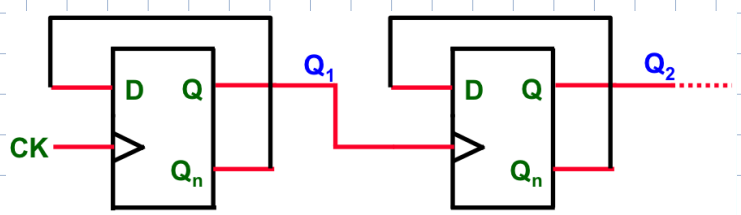
Time Constraints

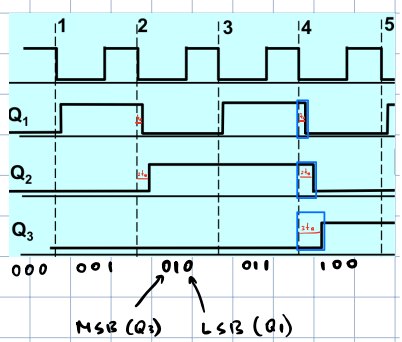


usually $t_H < t_{p\,CK\to Q}$

so we consider only t_p and t_{su}

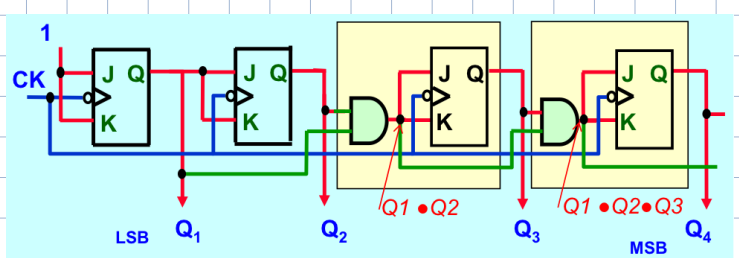
Asynchronous counter

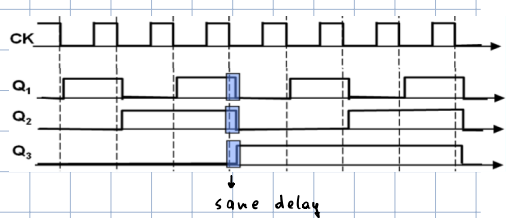




- problem → delay increases for every additional DFF
- given Q_1 with the delay T_{pd}, $T_{pdM} = M \cdot T_{pd}$

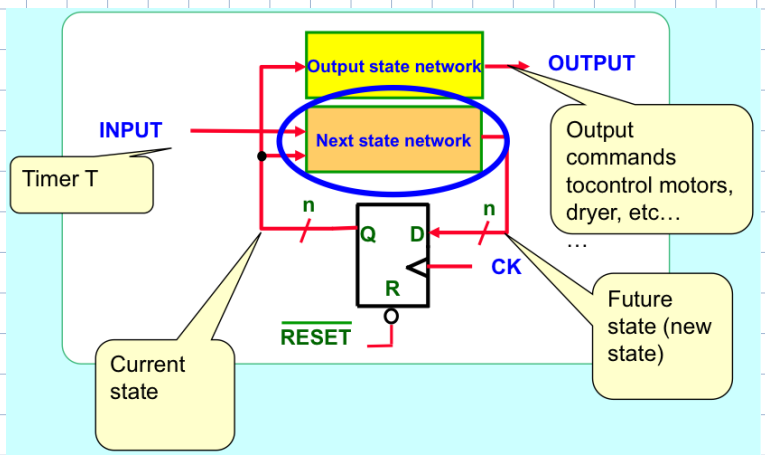
Synchronous Counter





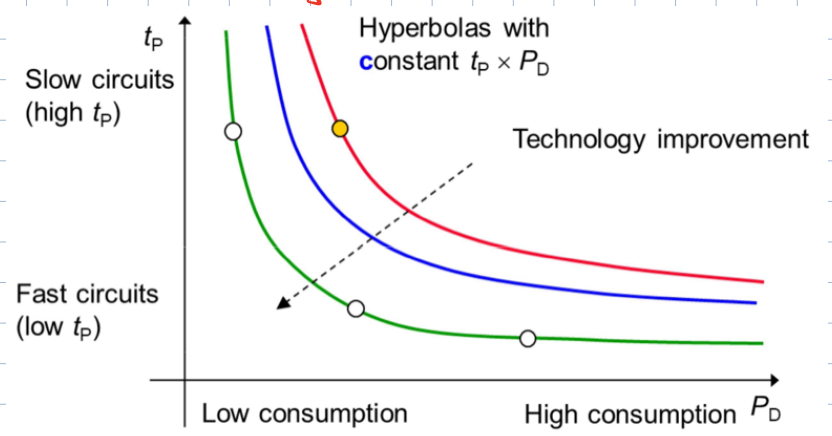
Max frequency: $T_{total} = T_{CK\to Q} + 2\,T_{log} + T_{su}$

Finite State Machine



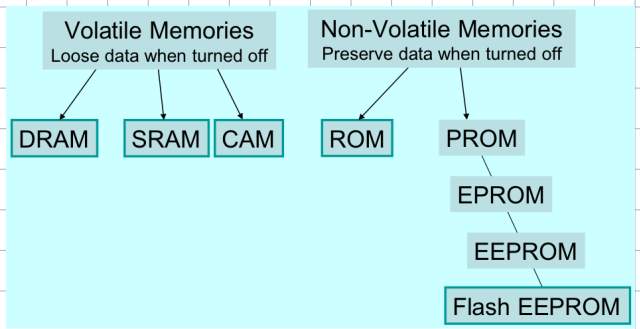
FSM is also a sequential circuit !

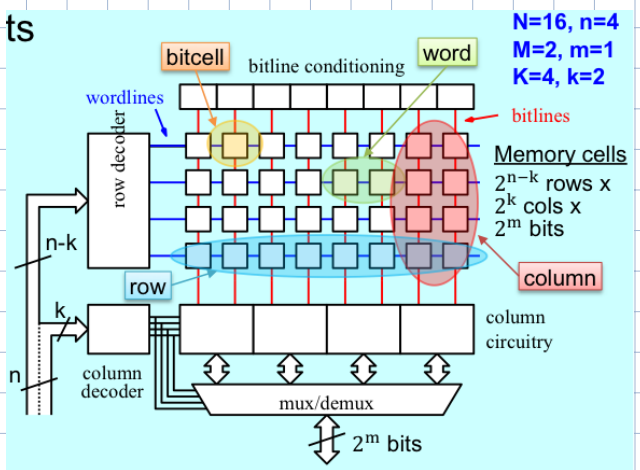
Power-delay chart



min{product(T,P)} = best performance

Memories

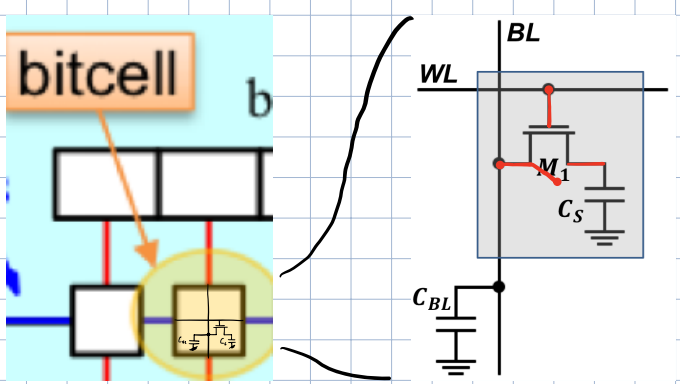




→ $N = 2^n$ → total # of words
→ $M = 2^m$ → # of bitcells that a word consists of
→ $K = 2^k$ → # of columns of words

n bits address
- n-k → row address
- k → column address

DRAM



C_s → Storage capacitor
↳ switch is controlled by WL

C_{BL} → Paracitic capacitor

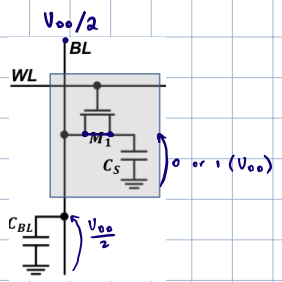
$C_s << C_{BL}$

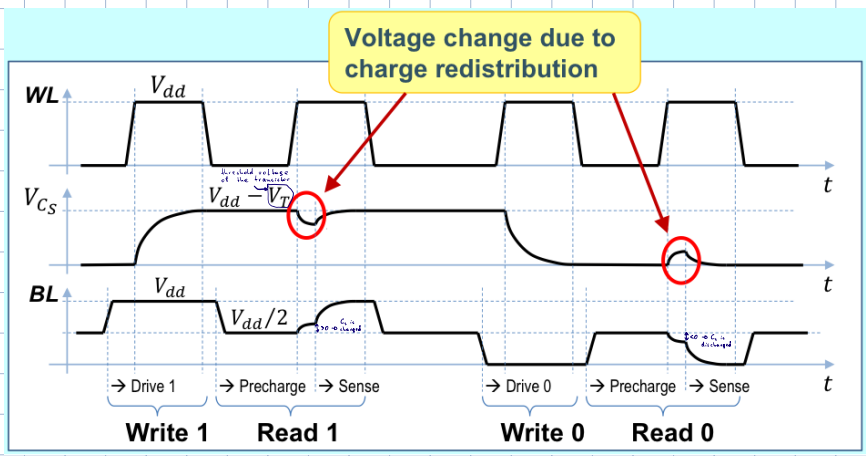
Write:

$W_L = 1$
- $BL = 1$ → C_s charged (write 1)
- $BL = 0$ → C_s discharged (write 0)

Read:

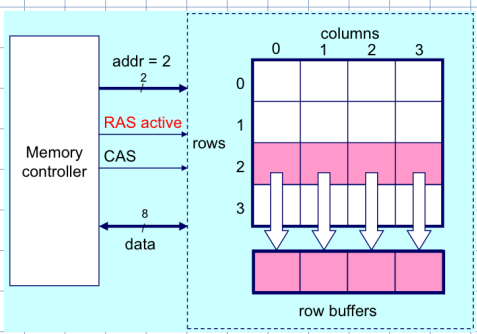
1. BL is pre-charged at half supply voltage ($V_{dd}/2$)
2. Select row with WL, charge transfer $C_S \leftrightarrow C_{BL}$
   - Small ΔV_{BL} positive (read 1) or negative (read 0)
3. ΔV_{BL} amplification → *sense amplifier*
4. Re-write the bit to restore the C_S voltage

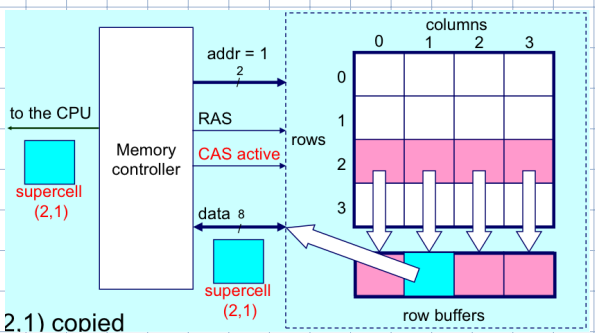




Read / write graph

Row - Column memory access:





2.1) copied

RAS → select row #
Move the row to row buffer

CAS → Select column #

* In DRAM (Asynchronous DRAM),

RAM and CPU clock are not synchronized

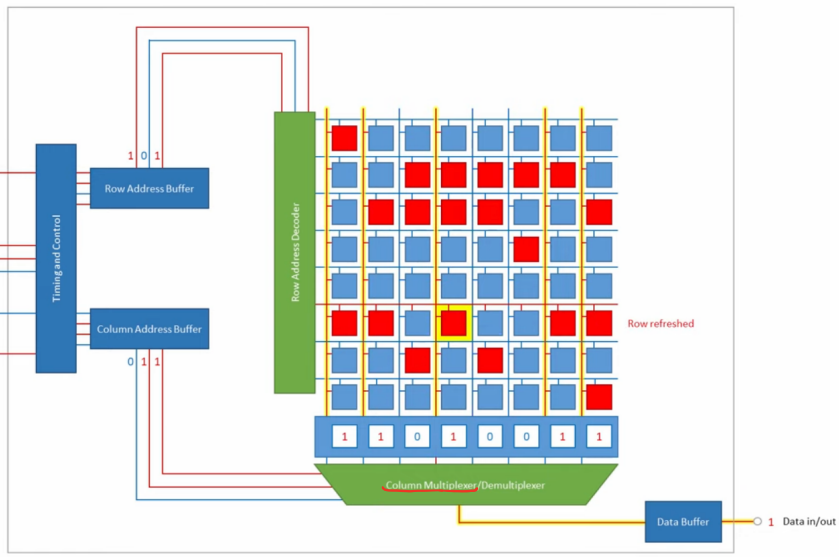
Read Cycle

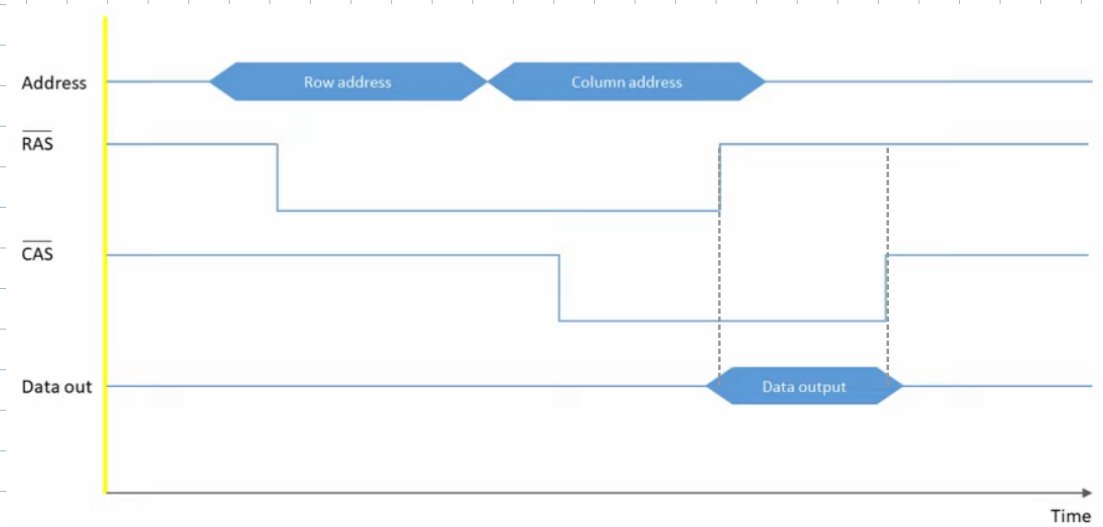
row address enable

column address enable

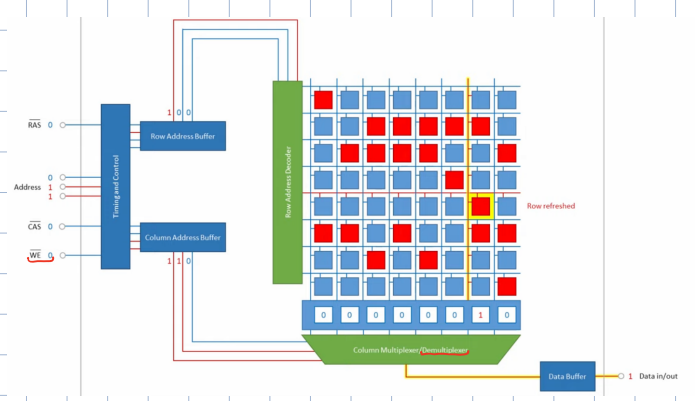
write enable

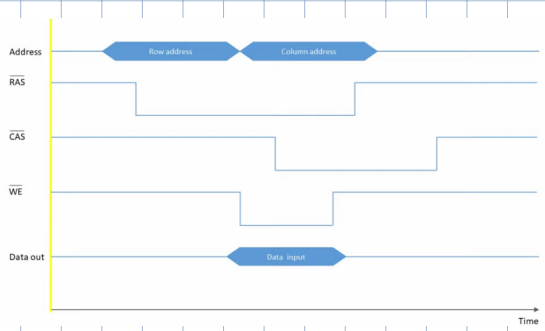
all on the falling edge (0)



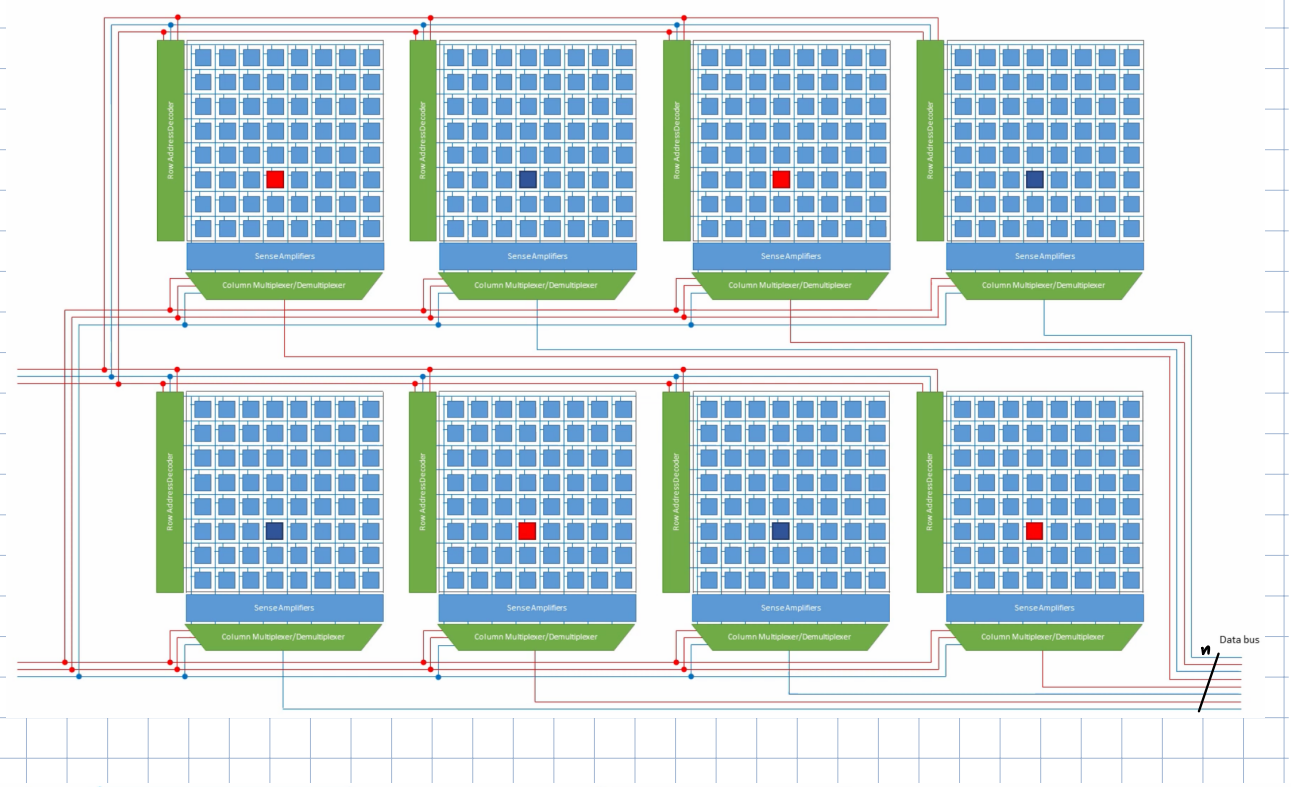


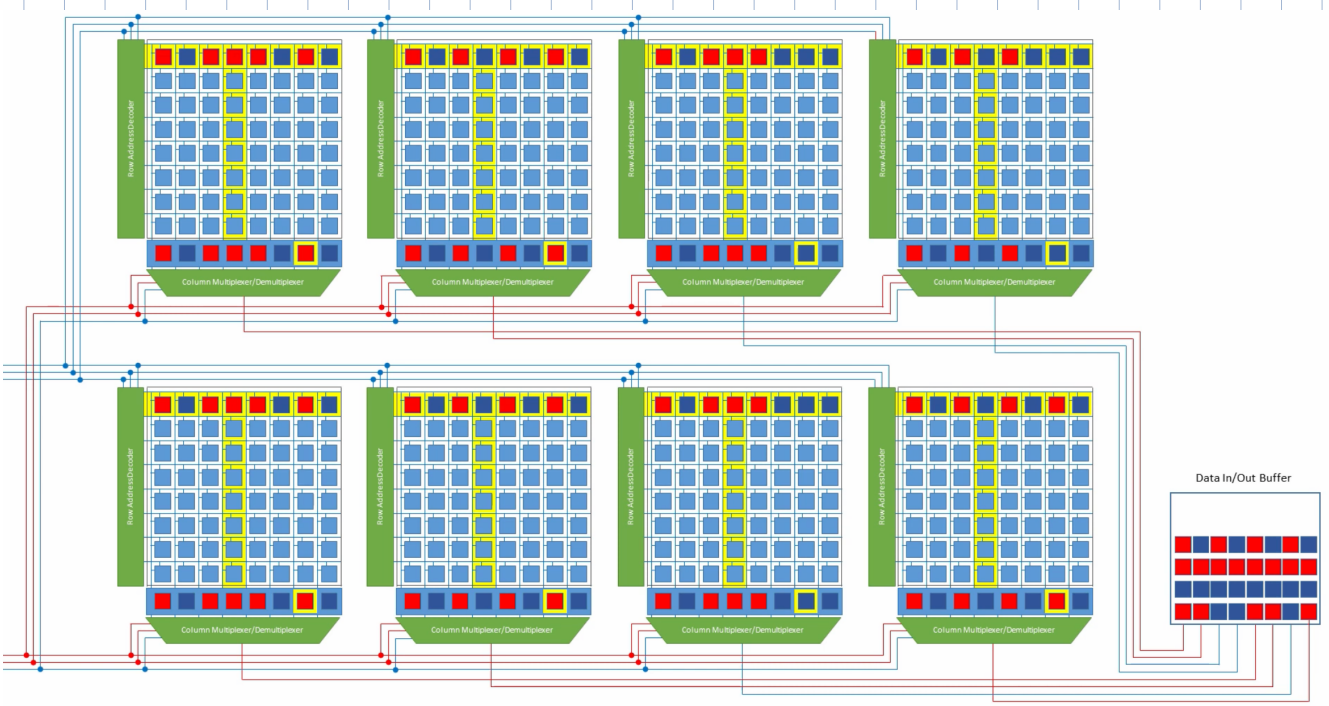
Write Cycle





n - bit DRAM





When we have a line read on each dram, instead if reading a single column and changing the row, we can read multiple columns on a single row. The number of columns that can be read on a single row is named ***Burst Length.***

Types of DRAM

DDR:

- Operations on both the **rising and falling clock edges**
- 2x faster than SDRAM
- Higher power consumption

Maximum speed / bandwidth
=
bus width (bits)
×
2 × bus clock frequency

SDRAM:

- Reads and writes on the rising edge of the clock signal

Maximum speed / bandwidth
=
bus width (bits)
×
bus clock frequency

ex

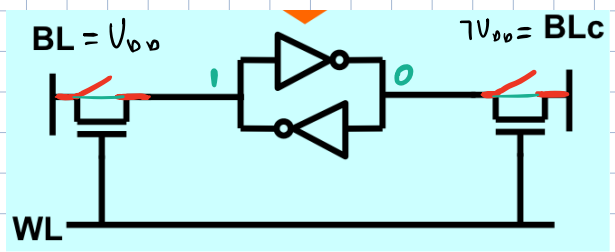
Maximum speed of 8-bit DRAM DDR3 with bus clock at 1000 MHz is 16 Gbit/sec (8bit · 2 · 1000 MHz)

DDR SDRAM Standard	Bus clock (MHz)	Internal rate (MHz)	Prefetch (min burst) (ns)	Transfer Rate (MT/s)	Voltage (V)	DIMM pins	SO-DIMM pins	Micro-DIMM pins
DDR	100–200	100–200	2	200–400	2.5 / 2.6	184	200	172
DDR2	200–533	100–266	4	400–1066	1.8	240	200	214
DDR3	400–1066	100–266	8	800–2133	1.5	240	204	214
DDR4	1066–2133	200–400	8	2133–4266	1.05 / 1.2	288	256	-
DDR5	1600–3600	200–450	16	3200–7200	1.1	288	262	-

minimum burst length

SRAM → 6T-SRAM

2 inverters = 4 transistors
2 MOSFET = 2 transistors
} 6 trans



Write

1. BL forced to 1 (or 0) and BLc to 0 (1)
2. Selection via WL = 1

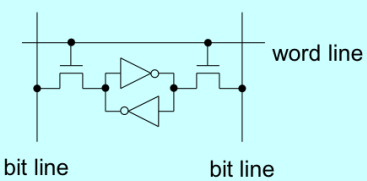
Read

1. BL & BLc pre-loaded to V_{DD}
2. Selection via WL = 1
3. BL or BLc lowered, depending on the stored bit
4. Activation of the Sense Amplifier

if BLc gets reduced → stored bit is 1
if BL gets reduced → stored bit is 0

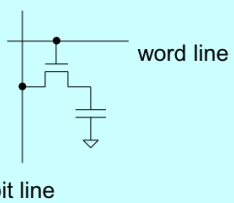
Comparison:

SRAM



- Larger cell ⇒ lower density, higher cost/bit
- Very low static consumption
- Non-destructive reading
- No refresh
- Simple reading ⇒ faster access
- Standard manufacturing process ⇒ natural choice for integration with logic circuits

DRAM



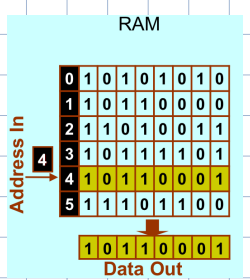
- Smaller cell ⇒ higher density, lower cost/bit
- Needs periodic refresh **and** refresh after each reading
- Complex reading ⇒ slower access
- Specific manufacturing process ⇒ difficult integration with logic circuits
- High density implies non-trivial addressing schemes

	SRAM (Static RAM)	DRAM (Dynamic RAM)
Usage	Cache Memory	Main Memory
Speed	Very Fast	Fast
Cost	Costly	Cheaper than SRAM
Density	Low	High

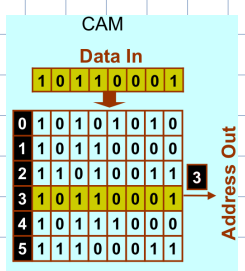
bcs. we need refrash after after write

mem. cells / unit area

CAM

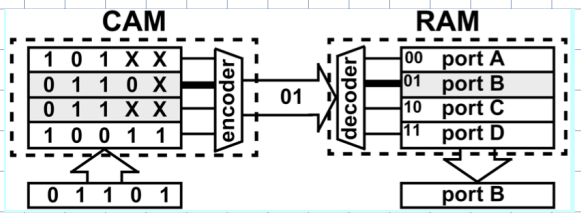


VS



give the data and get the address of the data

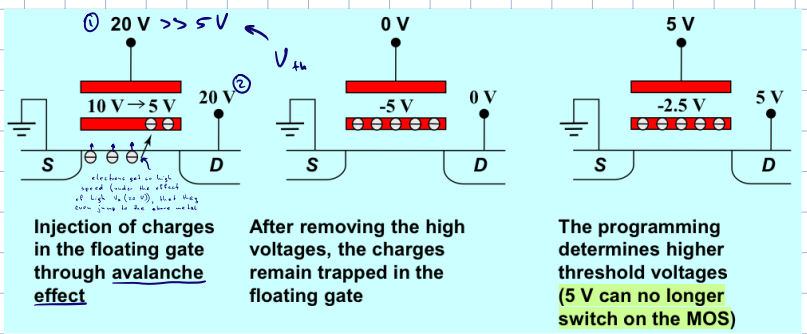
example → usually used with RAM



where the forwarding table is →

Entry No.	Address (binary)	Output port
1	101XX	A
2	0110X	B
3	011XX	C
4	10011	D

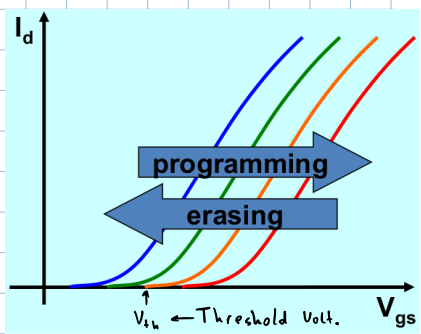
FAMOS — Floating-Gate MOS ← 1st method of Programmable mem.



Injection of charges in the floating gate through avalanche effect

After removing the high voltages, the charges remain trapped in the floating gate

The programming determines higher threshold voltages (5 V can no longer switch on the MOS)



EPROM → Based on FAMOS

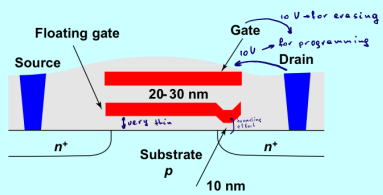
- Storage permanence
  - Charges in the floating gate slowly lost due to leakage currents
  - Programming state lost after 10s of years
  - Erasing and programming restores the charges

- Number of Program/Erase cycles
  - The lattice is damaged by programming and erasing
  - The number of P/E cycles is limited
    - 10000 – 100000 times
    - Afterwards the memory starts showing failures
  - Reading has no side effects
    - Frequent reads, rare writes

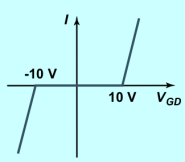
2nd method

Flotox-type EEPROM

- Electrically Erasable and Programmable ROM
  - Programming mechanism is reversible



FLOTOX Transistor

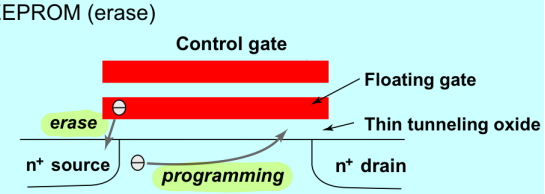


I-V characteristic Fowler-Nordheim

3rd method

Flash-type EEPROM

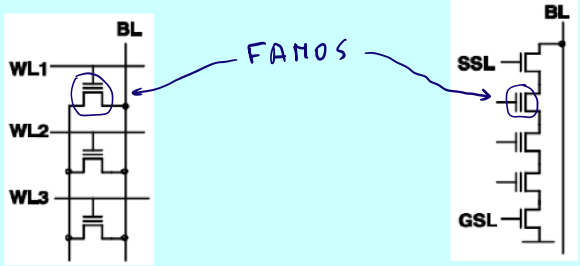
Combines
- EPROM (programming)
- EEPROM (erase)

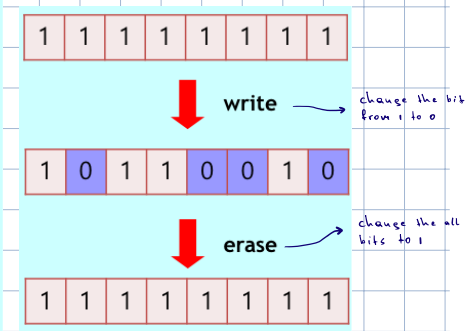


Flash Memory

- NOR flash ⟵ vs ⟶ • NAND flash
  - ✓ Random access
  - ✓ Fast read (random access)
  - ✓ Slow write and erase
  - ✓ Used mostly for code (random access)
  - ✓ Page access
  - ✓ High density, lower cost
  - ✓ Faster write than erase
  - ✓ Used mostly for data with sequential access

it reads the whole row, when it wants to access a single word





NOR (Code): fast read, slow write

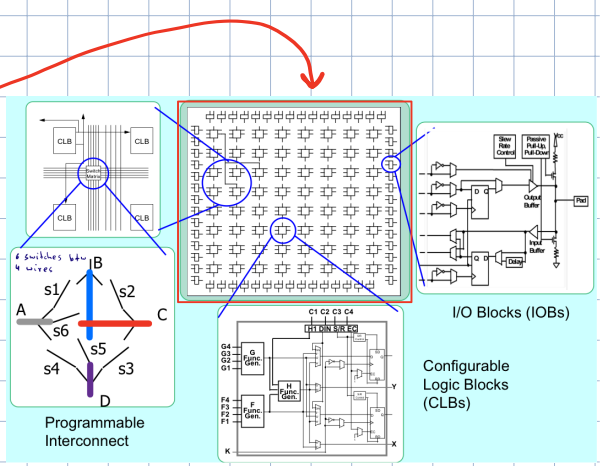
NAND (Data): fast write, lower cost

Basic operations:

- ♦ Page
  - Read / Write (in NAND flash)
- ♦ Block
  - Erase

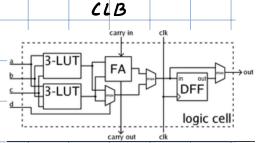
Programmable Logic

FPGA



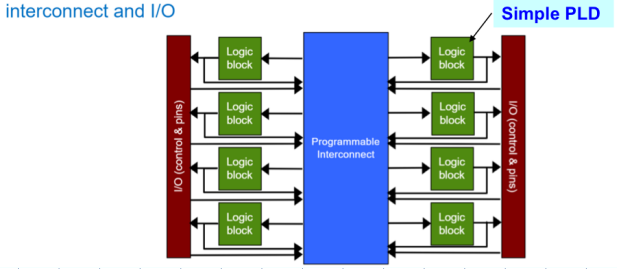
FPGAs consist of:
- CLBs
- IO blocks
- programmable switches

CLBs consist of:
- LUT
- MUX
- D-FF
- FA (Full adder)



CPLD

Combine multiple PLDs (logic blocks) in single device with programmable interconnect and I/O



Each PLD consists of:
- Logic blocks
- D-FF
- MUX

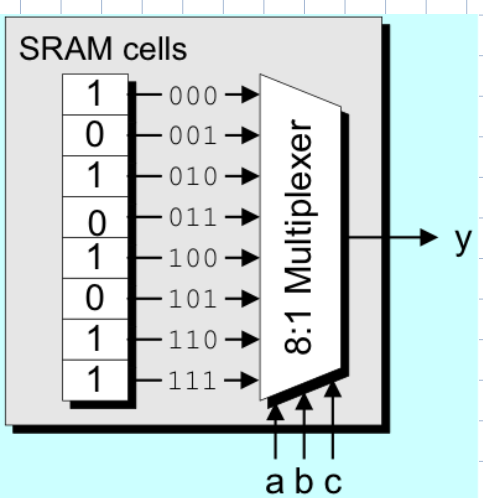
- FPGA and CPLD can be programmed with the HW Languages such as Verilog or VHDL

Moore's Law

It states that the number of transistors on a microprocessor chip will double every two years or so — which has generally meant that the chip's performance will double, too.

LUT

an example of LUT to implement $y = ab + c'$ (3 inputs)



→ 8 SRAM cells
1 MUX

Verilog HW

Verilog is intrinsically a parallel language (to model HW)

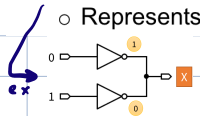
- Gate level, structural, RTL modeling
  - All instructions are concurrent
  - The order of the instructions is NOT important
- Behavioral
  - Instructions are executed sequentially
  - The order of the instructions is important

- Register Transfer Level (RTL) and Behavioral modelling
  - wire and reg
  - Continuous assignment ← RTL
  - always block ← Behav. modeling
- Design flow for FPGA and ASIC design

Data types

- wire — stateless (does not store a value)
- reg — stores the last value assigned

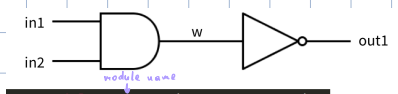
- 4 default values
  - Value **0**
    - Represents a logic zero, or false condition
  - Value **1**
    - Represents a logic one, or true condition
  - Value **Z** or **z**
    - Represents a high impedance value
  - Value **X** or **x**
    - Represents an unknown (undefined) logic value



- Constants → [size]['base]value
  - 8'b1011_1101 // 8-bit binary
  - 'hA3F0 // 32-bit hexadecimal
  - 16'o56377 // 16-bit octal
  - 32'd999 // 32-bit decimal

- default 'base is decimal
- default size is 32 bits

• Logic Gate modeling




```
1  module my_gate (out1, in1, in2);
2
3  output out1;
4  input  in1, in2;
5
6      wire w;
7      and (w, in1, in2);
8      not (out1, w);
9
10 endmodule
```

port list

Each internal signal should be defined as wire
Ports are implicitly defined as wire

Logic gate primitive instantiation
- It looks like a function call, but it is different
- If repeated, multiple independent instances are included

By convention (opposite to C/C++)
- The first connection is the output
- The following connections are inputs

AND	0	1	X	Z
0	0	0	0	0
1	0	1	X	X
X	0	X	X	X
Z	0	X	X	X

OR	0	1	X	Z
0	0	1	X	X
1	1	1	1	1
X	X	1	X	X
Z	X	1	X	X

XOR	0	1	X	Z
0	0	1	X	X
1	1	0	X	X
X	X	X	X	X
Z	X	X	X	X

NAND	0	1	X	Z
0	1	1	1	1
1	1	0	X	X
X	1	X	X	X
Z	1	X	X	X

NOR	0	1	X	Z
0	1	0	X	X
1	0	0	0	0
X	X	0	X	X
Z	X	0	X	X

XNOR	0	1	X	Z
0	1	0	X	X
1	0	1	X	X
X	X	X	X	X
Z	X	X	X	X

BUF	
0	0
1	1
X	X
Z	X

NOT	
0	1
1	0
X	X
Z	X

• Delays

Delays are expressed as **#(Tp)** or **#(Tplh, Tphl)** where each delay is:
- Either a single value, the maximum **max**
- Or three values, **min:typ:max**

```
xor #(2:3:4, 5) (s, x, y);
and #(3.567) (co, x, y);
```

Tplh Tphl

Continuous Assignment

Alternative higher Abstraction level to describe combinational logic

• Syntax

assign [#delay] <net_name> = <expression>;
The destination of an assignment should be a **wire** or an output port

• Examples

```
assign out = a & b | c;        // boolean or arithmetic expression
assign eq = (a + b == c) ;     // adder and comparator
wire #10 inv = ~in ;           // inverter with a 10 time units delay
wire [7:0] c = a+b ;           // 8-bit adder
```

• Operators:

Arithmetic Operators	
a + b	Add
a - b	Subtract
-a	Negate
a * b	Multiply
a / b	Divide
a % b	Remainder

Bitwise Operators	
~a	bitwise NOT
a & b	bitwise AND
a | b	bitwise OR
a ^ b	bitwise XOR
a ~^ b	bitwise XNOR
a ^~ b	bitwise XNOR

Shift Operators	
a << n	Logical shift left
a >> n	Logical shift right
a <<< n	Arith shift left
a >>> n	Arith shift right
{a, b}	Concatenate
~^a	XNOR all bits

Relational Operators	
a == b	Equality
a != b	Inequality
a < b	Less than
a > b	Greater than
a <= b	Less or equal
a >= b	Greater or equal

Reduction Operators	
&a	AND all bits
|a	OR all bits
^a	XOR all bits
~&a	NAND all bits
~|a	NOR all bits
~^a	XNOR all bits

Logical Operators	
!a	Negation
a && b	Logical AND
a || b	Logical OR
sel?a:b	Conditional

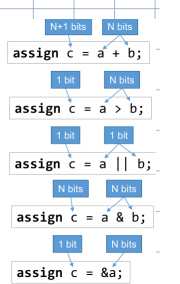
Arithmetic	
+ -	plus and minus

Relational	
> >=	greater, greater or equal

Logical (like C/C++)	
&& ||	logical AND and OR

Bitwise	
&	bitwise AND

Reduction	
& ~&	AND and NAND reduction



All bits of a are ANDed together and the result assigned to c

Example:

the value of "parameter"

actual names of the ports

formal names of the ports

param. value explicitly

param. takes the default value 2

```
1  …
2  adder #(8) adder_8 (co, sum, a, b, ci);
3  adder #(.width(8)) adder_8 (co, sum, a, b, ci);
4  adder adder_2 (.a(a), .b(b), .cin(ci), .cout(co), .s(sum));
5  adder adder_2 (.s(sum), .a(a), .b(b), .cin(ci));
6  adder #(2) adder_2 (, sum, a, b, ci);
7  …
```

```
1  module adder (cout, s, a, b, cin);
2
3      parameter           width = 2;
4      input  [width-1:0] a, b;
5      input               cin;
6      output [width-1:0] s;
7      output              cout;
8
9      assign {cout, s} = a + b + cin;
10
11 endmodule
```

Module name | Instance name | Parameter by position

Arguments by position

- Leaving a position empty makes the corresponding port not connected
- Port cout is not connected in this example

```
1  …
2      assign c = a & b;
3      assign d = a | c;
4      assign s = c ^ d;
5  …
```

⟷

```
1  …
2      assign s = c ^ d;
3      assign c = a & b;
4      assign d = a | c;
5  …
```

The order is not important!

Always Block

- Syntax →
  - `always @(sensitivity_list) begin … end` → it get triggered when any element in sensitivity list changes value
  - `always @(posedge clock)`
    `always @(negedge clock or reset)` → it gets triggered on the edge of clock
- Output should be of type "reg", NO "assign" keyword is needed
- Blocking and non-Blocking assignments
  - **Blocking** assignments has an immediate effect (like in C/C++)

```
1  always @(a or b) begin
2      a = b;
3      b = a;
4  end
```

After the execution of the always block
- a == 1, b == 1
- The always block is executed **twice**

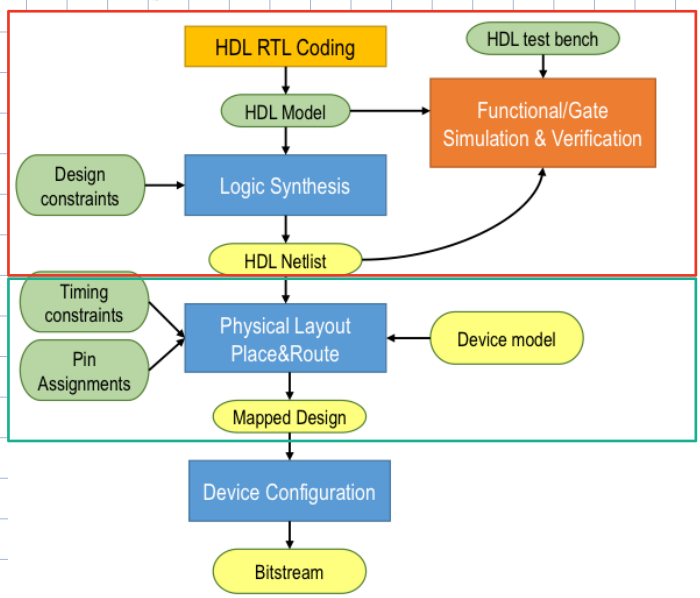
  - **Non-blocking** assignment occurs after some time has elapsed, it assigns a **future value**

```
1  always @(a or b) begin
2      a <= b;
3      b <= a;
4  end
```

After the execution of the always block
- a == 1, b == 0
- But then a == 0, b == 1
- But then a == 1, b == 0
- And so on…
- The always block is executed **infinite times**

- "if" and "case" statements can be used only inside "always" block

Design flow for FPGA design



Design flow step (user)

Design flow step (tool)

Design flow verification (tool)

Design files (user)

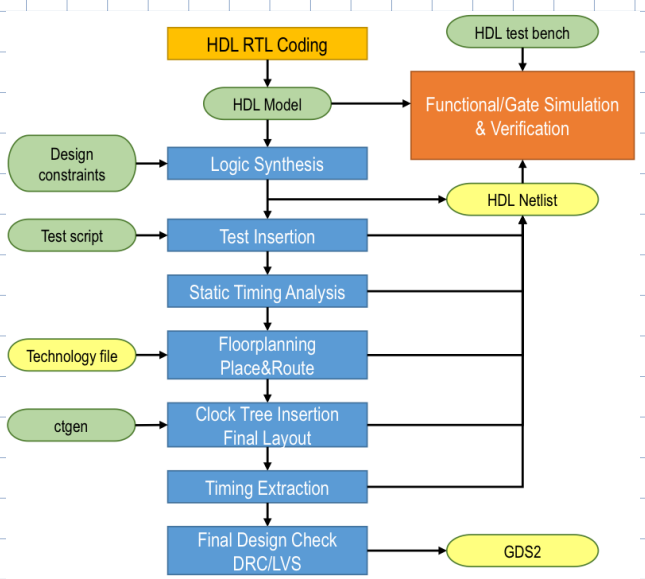
Design files (tool)

EDA Tools
- Intel/Altera Quartus Prime
- AMD/Xilinx Vivado
- Mentor Graphics Modelsim

- Compiles an RTL specification
  - Verilog or VHDL
  - Generates a netlist of registers and combinational logic circuits
- Optimizes the combinational logic circuits
  - Using Boolean algebra
- Generates a netlist of interconnected logic gates
  - Verilog, VHDL or EDIF
- Initial timing estimates
  - Ignores
    - LUT mapping
    - Place & Route
- May start from a (subset of) C/C++ specification
  - Higher level of abstraction

- Two phases
  - Placing and Routing
  - Distinct because otherwise the problem is too complex
- Placing
  - Assigns a **physical position** (CLB or LE) to each LUT and Flip-Flop in the design
  - Goal: minimize **distances** to optimize the delay
    - The distance is an **estimate** of interconnection lengths
    - Long interconnections have higher delays (higher time constant $\tau = RC$)
- Routing
  - Assigns a **sequence of interconnection segments** to each interconnection
    - From outputs to inputs of LUTs and FFs in the placed design
    - Chosen by properly opening and closing switches (transistors) during programming
  - Goal: minimize the **length** of the interconnections

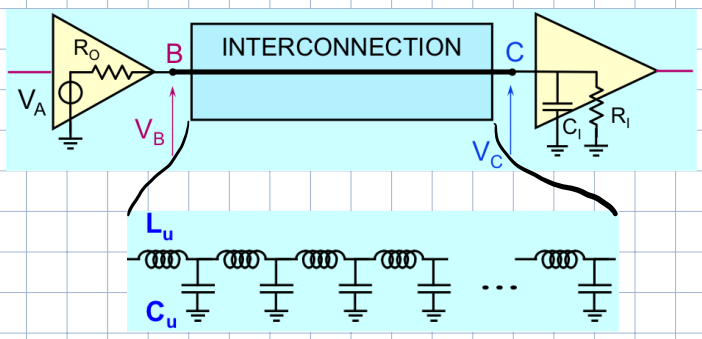
Design flow for ASICs



Comparison:

- The design flow steps are very similar to the FPGA design flow
- ASIC
  - Many more verifications
    - Masks are extremely expensive
    - Mistakes cannot be made!
  - Much greater variability of transistor and interconnection parameters
  - Many more design and development tools
    - Complex
    - Expensive
- FPGA
  - Much simpler
    - Manufacturing is already pre-characterized
    - Design tools are typically provided by the manufacturer
      - Cheap

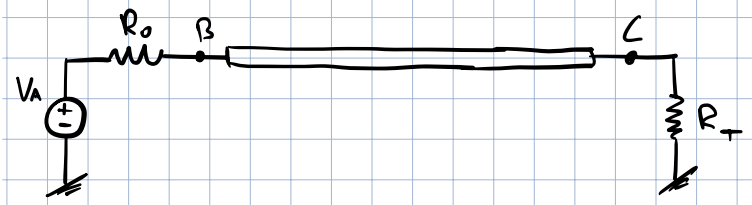
Interconnections



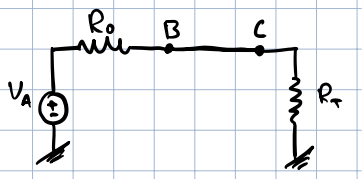
Parameters of interconnection

L_u → inductance per unit length
C_u → capacitance per unit length
L → length
u → propagation speed

$t_p = \frac{L}{u}$ → propagation time $\qquad Z_\infty = \sqrt{\frac{L_u}{C_u}}$ → characteristics impedance $\qquad u = \frac{1}{\sqrt{L_u C_u}}$



at steady state →

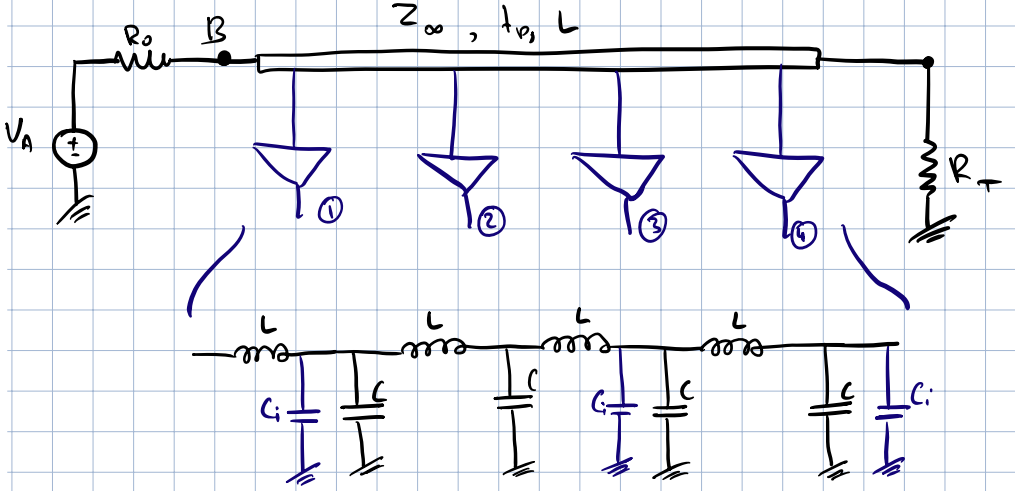


$$\Gamma_B = \frac{R_o - Z_\infty}{R_o + Z_\infty} \qquad \Gamma_T = \frac{R_T - Z_\infty}{R_T + Z_\infty} \qquad V_B(0) = \frac{Z_\infty}{R_o + Z_\infty} \cdot V_A$$

Types of terminations:

- Short Circuit
  - $R_T = 0 \rightarrow \Gamma_T = -1$
- Open Circuit
  - $R_T \approx \infty \rightarrow \Gamma_T = 1 \rightarrow$ backward voltage = incident voltage
- Load Matching
  - $R_T = Z_\infty \rightarrow \Gamma_T \approx 0 \rightarrow$ No back reflection

Multiple receivers along the transmission line



$$C_u' = C_u + \frac{C_i \cdot N}{L} \longrightarrow C_u \text{ has changed to } C_u'$$

$$Z_\infty' = \sqrt{\frac{L_u}{C_u}} \qquad Z_\infty' < Z_\infty$$

$$u' = \frac{1}{\sqrt{L_u \cdot C_u'}} \qquad u' < u \rightarrow t_p' > t_p$$

Incident Wave switching (IWS)

- All the receiver along the transmission line are turned on on the first incident wave

↳ $V_B^+(0) > V_{IH}$
↳ $R_T = Z_\infty$
} necessary conditions

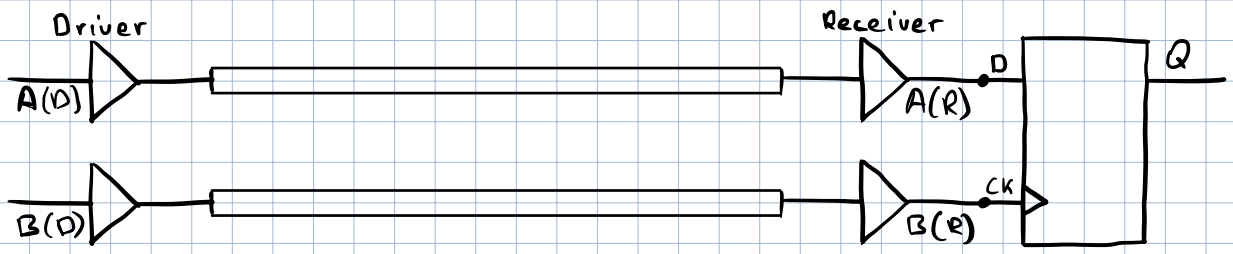
Reflective wave switching (RWS)

- The receivers over the transmission line don't turn on on the first incident wave and require a reflection wave

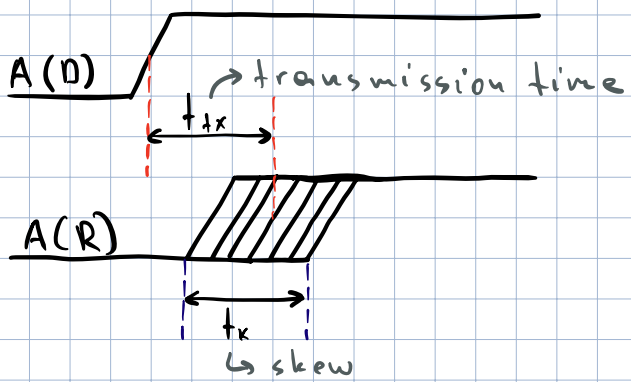
↳ $V_B^+(0) \leq V_{IH}$
↳ $V_B^+(0)[1 + \Gamma_T] \geq V_{IH}$
} necessary conditions

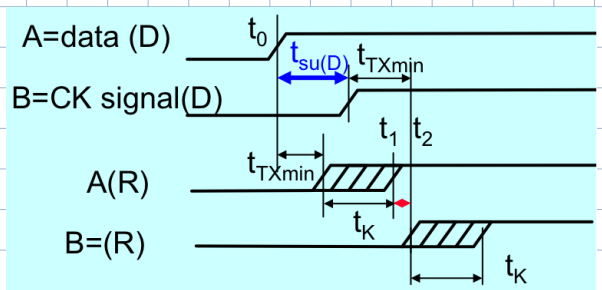
↳ $\Gamma_B > 0$
↳ $\Gamma_C > 0$

Synchronization Cycles



- D should not change during the setup time and hold time of the clock (CK)
- So we need to find a way of synchronizing the clock with respect to "D"



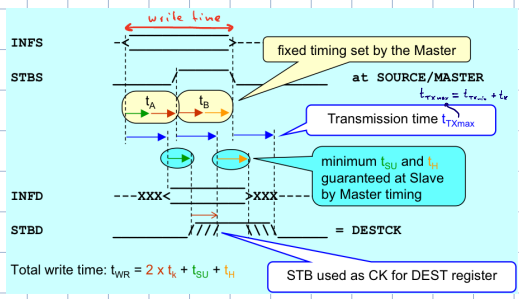


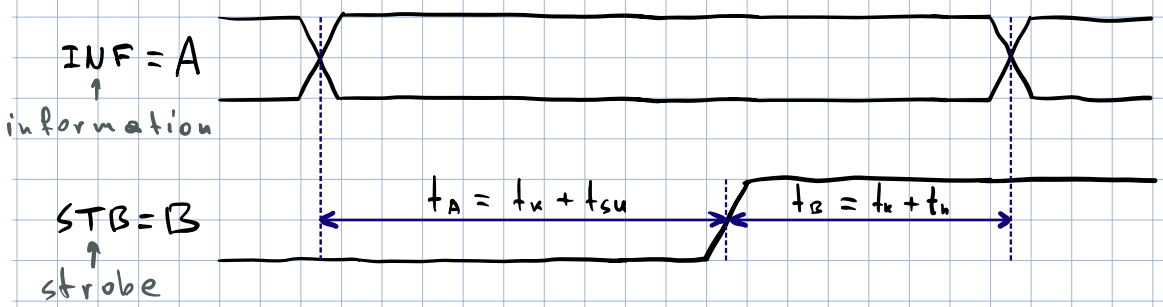
if $t_2 - t_1 < t_{su(D)}$ set-up constraint at the receiver is not satisfied

2 ways of solving this issue:

Synchronous cycle:

- All timing parameters are controlled by the source, following the sequence:
  - Send information (INF)
  - Wait t_A to guarantee t_{SU} $t_A \geq t_{SU} + t_K$
  - Send STB command (STB is the clock for destination register)
  - Wait t_B to guarantee t_H $t_B \geq t_H + t_K$
  - Remove INF and STB



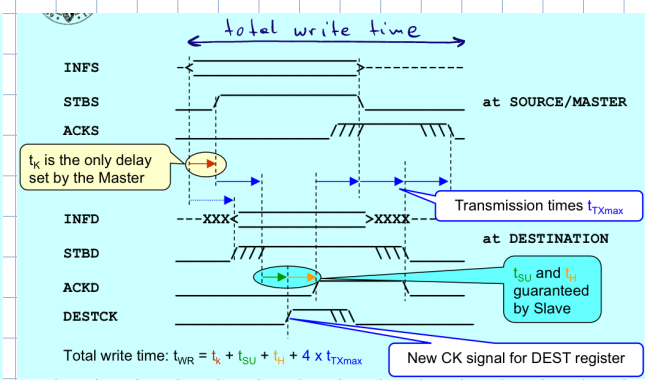


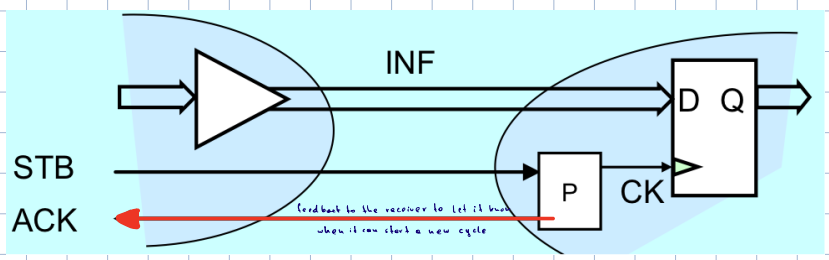
Total write time
Minimum cycle duration
$t_{WR} = t_H + t_{SU} + 2t_K$

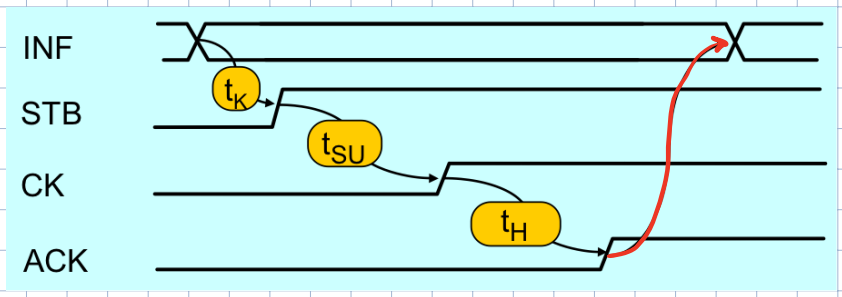
Asynchronous cycle:

- Source must "know" only t_K
- t_{SU} and t_H are inserted at destination.

  The sequence
  - Send information (INF)
  - Wait t_K
  - Send STB command
  - After the STB at destination generate the CK
  - Generate the ACK at destination
  - Wait for confirmation from destination (ACK, arriving after t_C)
  - Remove INF and STB







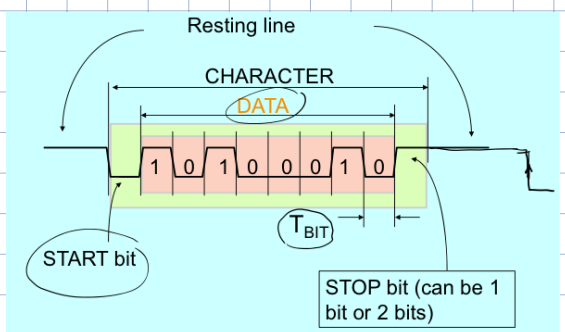
Total write time
=
Minimum cycle duration

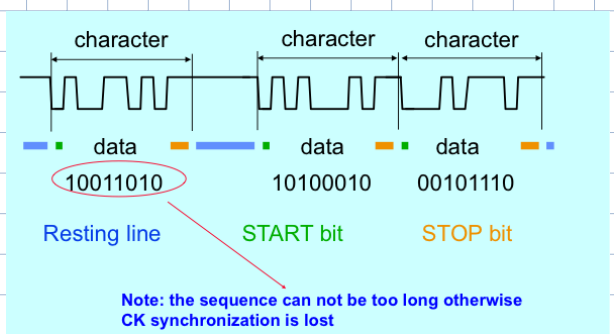
$$t_{we} = t_K + t_{su} + t_H + 4 \cdot t_{TX\,max}$$

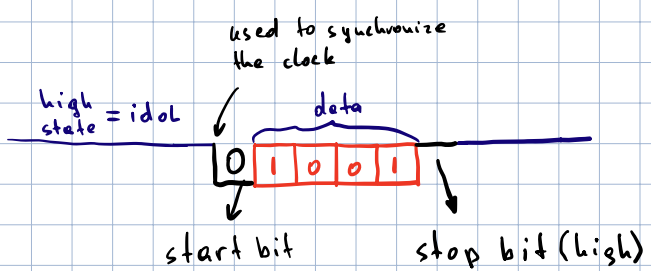
Bit synchronization

- Asynchronous Links
- Synchronous Links
  - Bipolar RZ
  - MLT
  - Manchester coding
  - Bit stuffing
  - xbyb

Asynchronous Links







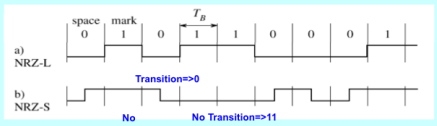
- Clock is generated at the receiver at the falling edge of start bit

Synchronous links

- Clock is not generated at the receiver, so we send 2 info:
  - data
  - clock

NRZ encoding

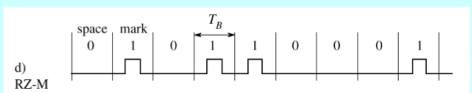
- NRZ: Not Return to Zero
  - Symbol may end with any state
  - NRZ-L: 1 represented by a H state, 0 → L state
  - NRZ-M: 1 represented by a transition (NRZ-S: 0=trans)



- No transition with fixed sequences
- Bandwidth: 1 transition/bit → Fmax = BitRate/2

RZ encoding

- RZ: Return to Zero (unipolar)
  - All symbols start and finish with 0 V (or other fixed state)
  - RZ-M: 1 represented by a pulse H (Mark) (RZ-S: 0 = H)



- No transition for constant sequences (0 per M)
- Bandwidth: 2 transitions/bit → Fmax = BitRate

MLT-3 encoding

- MultiLevel Transmission encoding, 3 levels: +, 0, -
  - 0: no change; 1: state change, with the following rules
  - Ternary code

- Bandwidth: Fmax = BitRate/4
- Used for Ethernet 100Base-TX

Manchester codes
(Embedded clock encoding)

- Phase modulation (π phase steps; Bi-Phase L)
  - 0: transition L → H
  - 1: transition H → L

- One transition/bit (at least) → self synchronizing:

This is because in each bit I have a transition (an edge) and I can synchronize the CK respect to this edge.

- Bandwidth: → Fmax = BitRate

4B5B example

- From 4-bit (16) to 5-bit (32)
- Redundancy used for:
  - Transitions insertions
  - DC removal:

    It means for example that 1111 is replaced With 111101
  - Control characters:
    ("free" 5-bit codes)
    11000: start 1
    10001: start 2
    00100: halt
    …

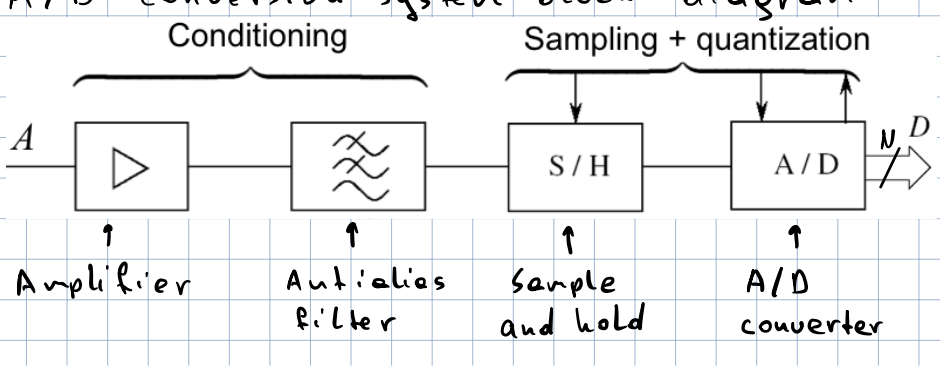
4B	5B	4B	5B
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

- The new blocks guarantees
  - » Presence of transitions
  - » DC level = 0 (or almost 0)
  - » Reserved codes for commands/controls
- Can be codified in NRZ (better use of bandwidth)
- Better band occupancy (% = increase in bit number)

Data Acquisition System

ADC

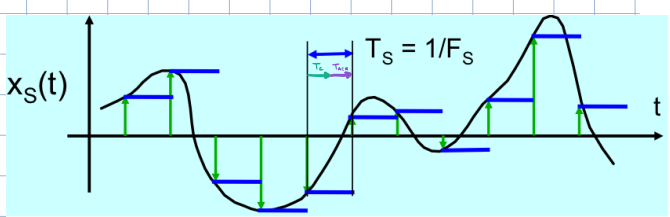
A/D conversion system block diagram



1. Protection circuit to limit the Analog signal voltage in the "safe" input range
2. Conditioning amplifier to adapt the input dynamics to the ADC converter dynamics and thus to reduce the SNR associate to the quantization error
3. Antialiasing filter to limit the Analog signal bandwidth and thus to reduce the aliasing error
4. Multiplexing to use the same S/H and ADC blocks for several channels
5. S/H unit to sample at discrete instant of time the Analog signal
6. ADC converter to convert the discrete Analog values in Digital values

Sample / Hold

Two operations required
- Sample: read the analog signal value at as specific time
- Hold: keep that value for some time
- Sample/Hold (or Track/Hold) unit

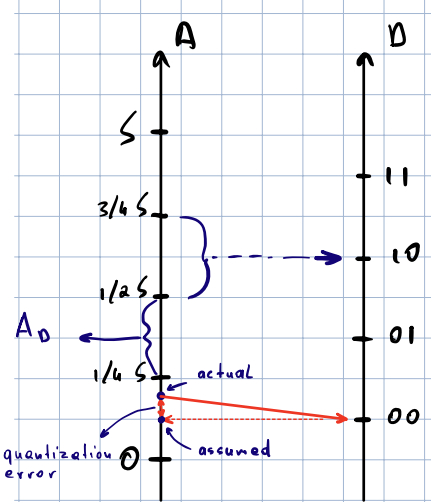


$T_s = T_c + T_{aca}$

$T_s \rightarrow$ Sampling time
$T_c \rightarrow$ Conversion time of ADC
$T_{aca} \rightarrow$ Acquisition time of data

Quantization

Example of 2-bit quantization



$$A_D = \frac{S}{2^N} \leftarrow \text{1 LSB}$$

$$|\varepsilon_q| \leq \frac{A_D}{2} = \frac{S}{2^{N+1}} \leftarrow \text{max. quantization error}$$

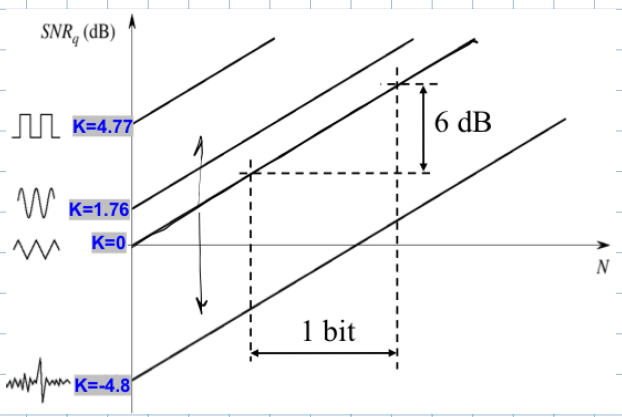
Signal to quantization Noise Ratio

$$SNR_{dB} = 10 \log_{10}\left(\frac{P_s}{P_N}\right)$$ ← Power of Signal / Power of Noise

$P_N > 0$

$$P_N = \frac{A_d^2}{12} = \left(\frac{S}{N}\right)^2 \cdot \frac{1}{12}$$ ← By increasing N, we can reduce P_N

$$SNR_q = (K + 6N)_{dB}$$ ← when $A = S$, where

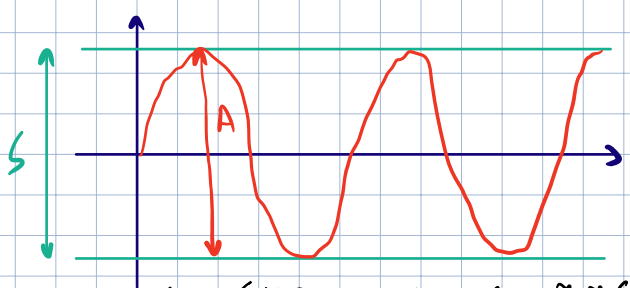


A → peak-to-peak voltage of the input

S → dynamic range of ADC

N → # of bits

ex

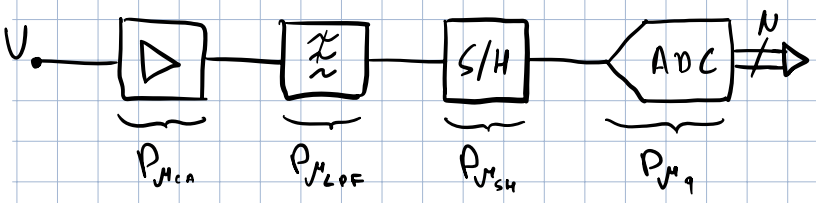


↳ $SNR_q = 1,76 + 6 = 7,76$ dB

if A < S

$$SNR_q = (K + 6N)_{dB} - 20 \log\left(\frac{S}{A}\right)$$

Total SNR_q



at every step we have noise

$$SNR_{tot} = \frac{P_s}{\sum P_{N_i}} \longrightarrow SNR_{tot\,dB} = 10\log_{10}\frac{P_s}{\sum P_{N_i}}$$

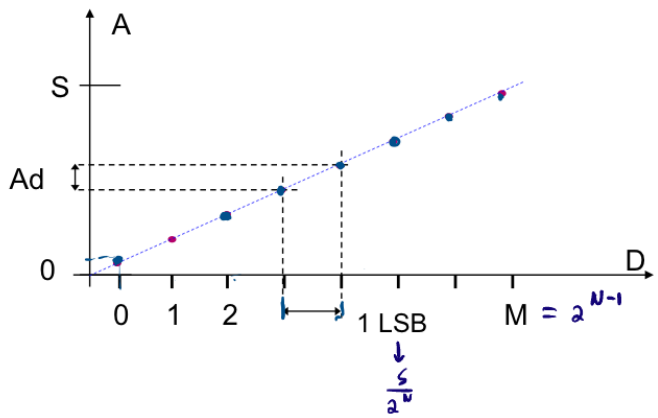
$$SNR_{tot} = (6 \cdot \underbrace{ENOB}_{\text{Effective \# of bits due to additional noise at each step}} + K)_{dB}$$

↳ $ENOB < N$

DAC

Input variable (D) is discrete
- The A(D) plot is a sequence of dots
- With constant Ad the dots are aligned

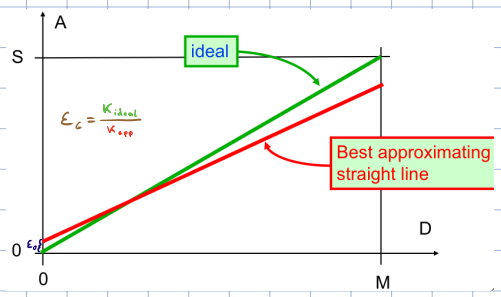
# of output voltages → 2^N



Errors:

- Linear
  - gain ε_G
  - offset ε_O
- Non-Linear
  - integral NL: ε_{INL}
  - differential NL: ε_{DNL}
- Dynamic params:
  - Settling time: t_s
  - Glitches

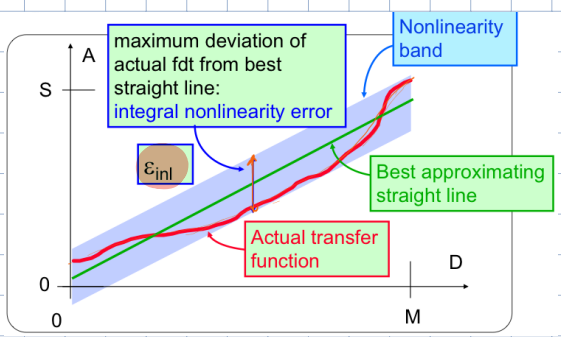
Linear errors

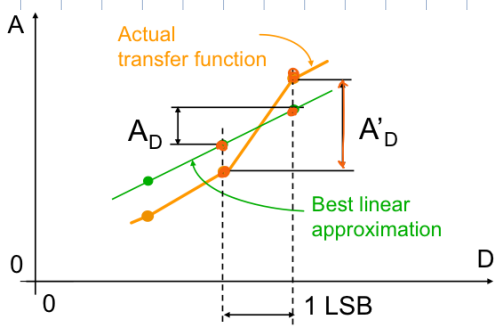


ε_0 → offset error (difference at 0)

ε_G → gain error (slopes ratio)

Non-linearity errors



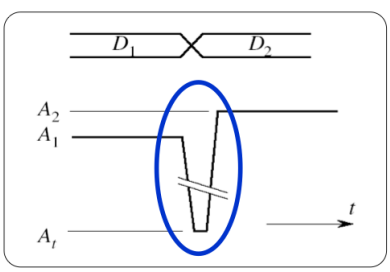


$\varepsilon_{DNL} = A_D - A_D'$

non-monotonicity ← $\varepsilon_{DNL} > 1\,LSB$ → slope inversion

Glitch

- In a transient the output can go to a widely wrong value (typically 0V or full-scale S).
- The spike is called GLITCH



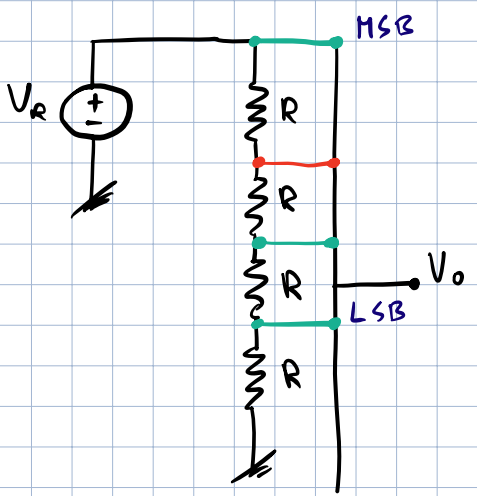
- it occures during the transition from one state to another

DAC

Uniform — Weighted resistor — Ladder network

example of 1011

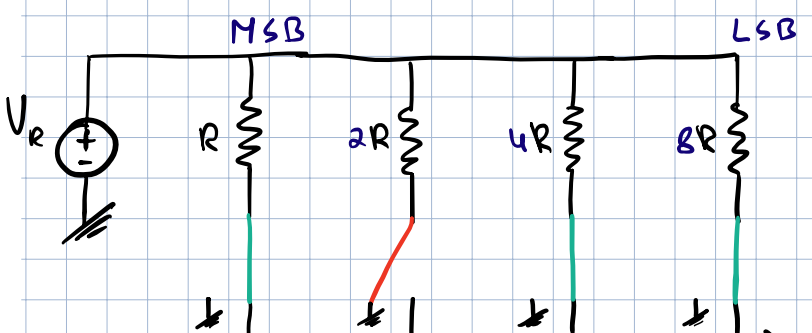
Uniform



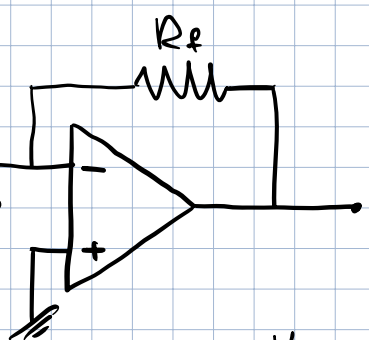
disadvantage:

N bits $\longrightarrow 2^N$ resistors

Weighted



$$V_{u\,MSB} = V_R \frac{R_f}{R} = \frac{5}{2}$$



adv: less # of resistors

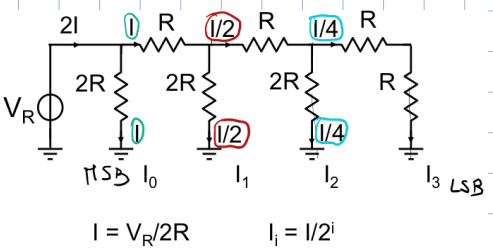
disadv: different values of resistors

MSB branch errors are differential non-linear error

$$I_o = \frac{V_R}{R} + \frac{V_R}{4R} + \frac{V_R}{8R}$$

$$V_o = -I_o \cdot R_f$$

Ladder Network

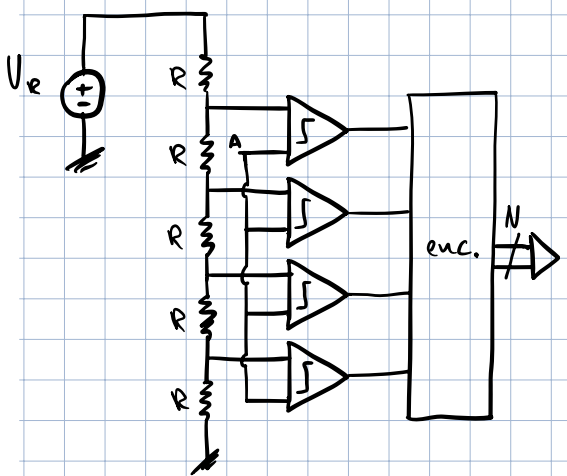


$I = V_R/2R$ $\quad I_i = I/2^i$

adv: easily scalable to any N

ADC

Flash



N bits

$2^N - 1$ comperators

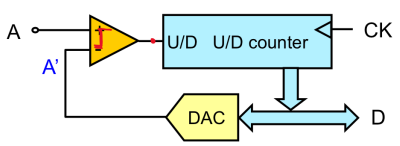
1 comparison cycle

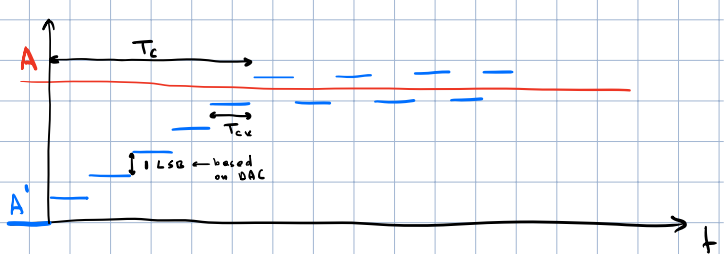
Fast → all comparators work simul.

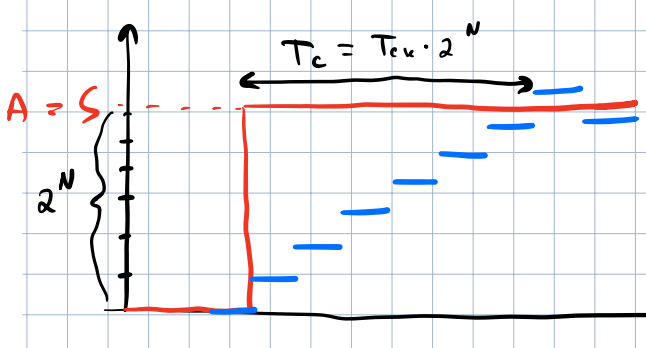
↳ $T_c = T_{comp} + T_{enc}$

Complex → many compar.

Tracking



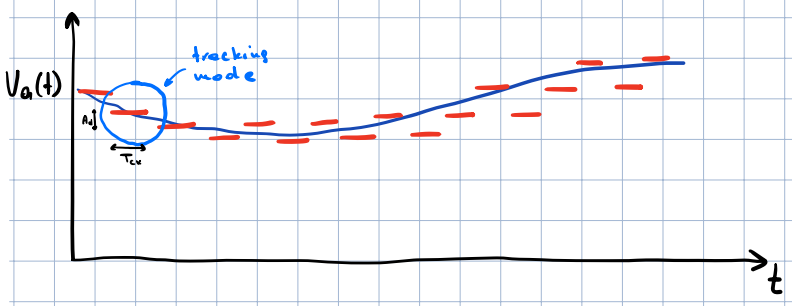




$T_{cmax} = T_{ck} \cdot 2^N$

tracking mode

↳ When the input changes very slowly, the conversion can track it



$$\Delta V_a \leq A_D \rightarrow \frac{dV_a}{dt} \cdot T_{ck} \leq A_D$$

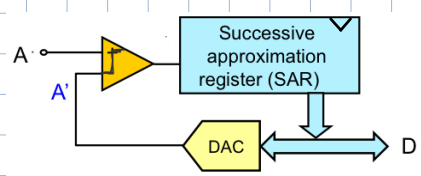
$$\frac{dV_a}{dt} \leq \frac{S}{2^N} f_{ck}$$

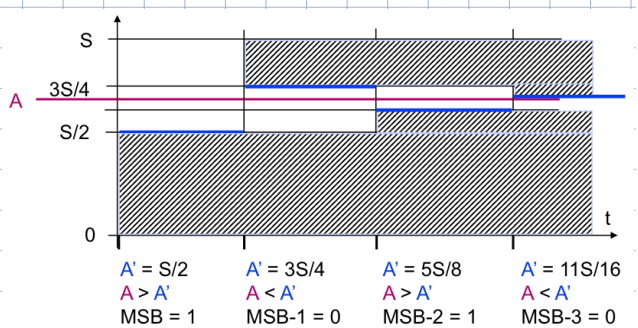
$T_c = T_{ck}$ ← in tracking mode

Slow → 2^N comparison cycle

Simple → 1 comparator

Successive Approximation





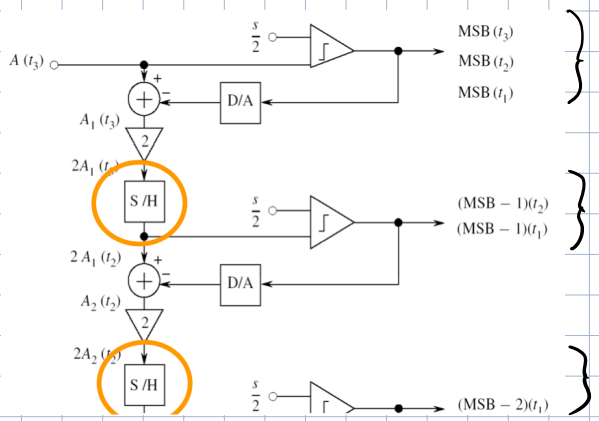
input A is compared with S/2:

A > S/2 → compare A with 3S/4
A < S/2 → compare A with S/4
...

Simpler → 1 Comparator

Faster → $T_c = N \cdot T_{ck}$ → N comparison cycle

Pipeline



Here each circuit can operate individually without waiting for the output of the prev. one

t_1 → Circuit 1 operates

t_2 → Circuit 1 and 2 operate simultaneously

t_3 → Circuit 1, 2 and 3 operate simultaneously

N comparators

1 Conversion Cycle

Summary

	Complex	Conv time	Latency
Parallel (flash)	2^N	$1 \cdot T_{ck}$	1
Pipeline	N	$1 \cdot T_{ck}$	N
Residue	N	$N \cdot T_{ck}$	N
Successive Approx	1	$N \cdot T_{ck}$	N
Tracking	1	$2^N \cdot T_{ck}$	2^N

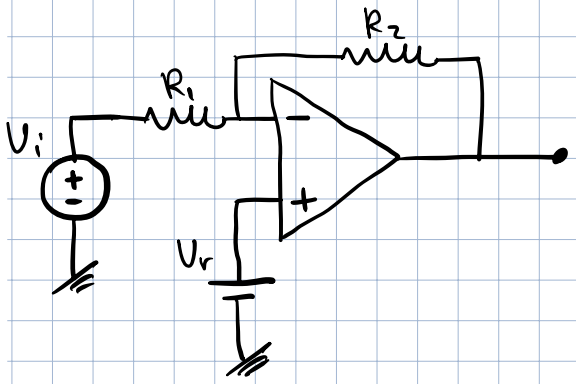
Complexity: proportional to the number of comparators.

Conversion time: the maximum number of comparator delay (clock periods) to complete a conversion

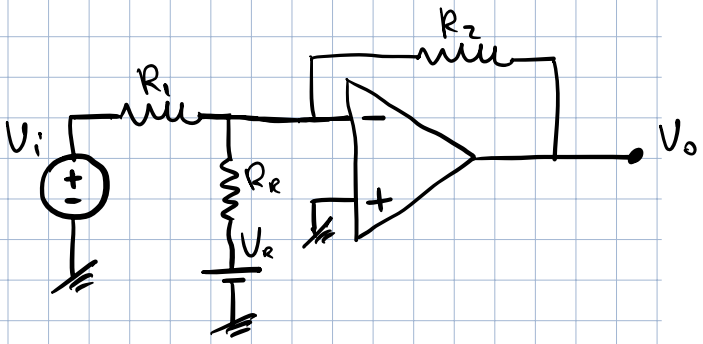
Latency: delay to get the ~~result~~ 1st sample

Inverting Amplifier with offset

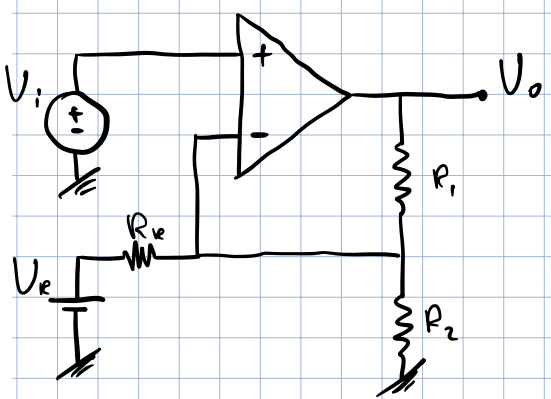
Positive offset



Negative offset

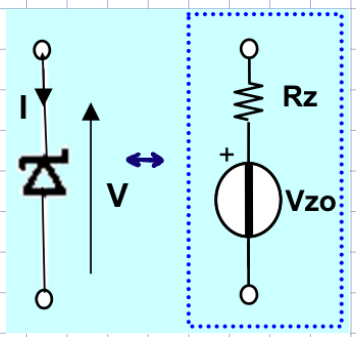


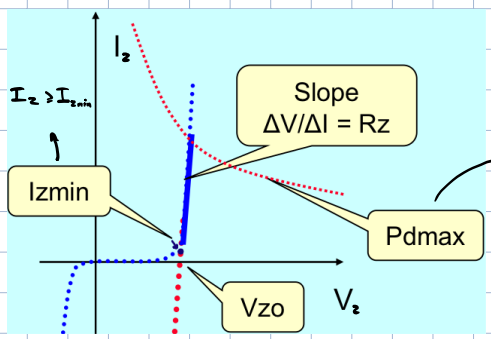
Noninverting Amplifier with offset



Power Circuits

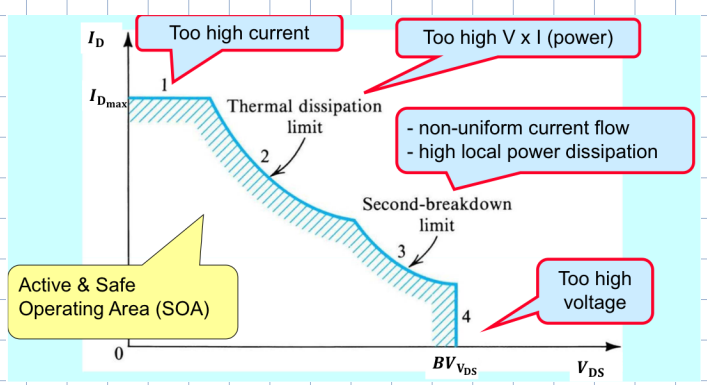
Zener diode





$P_d = I_z \cdot V_z \leq P_{d\,max}$

MOS operating limits

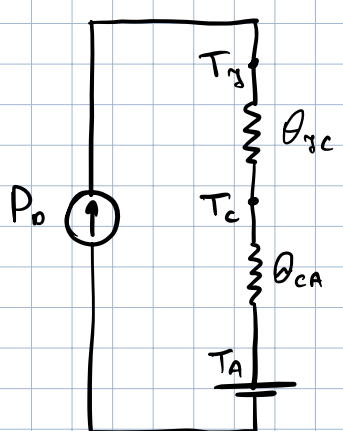


Power limit: $V \cdot I < P_{d\,max}$

Voltage: $V < V_{breakdown}$

Current: $I < I_{max}$

Thermal model

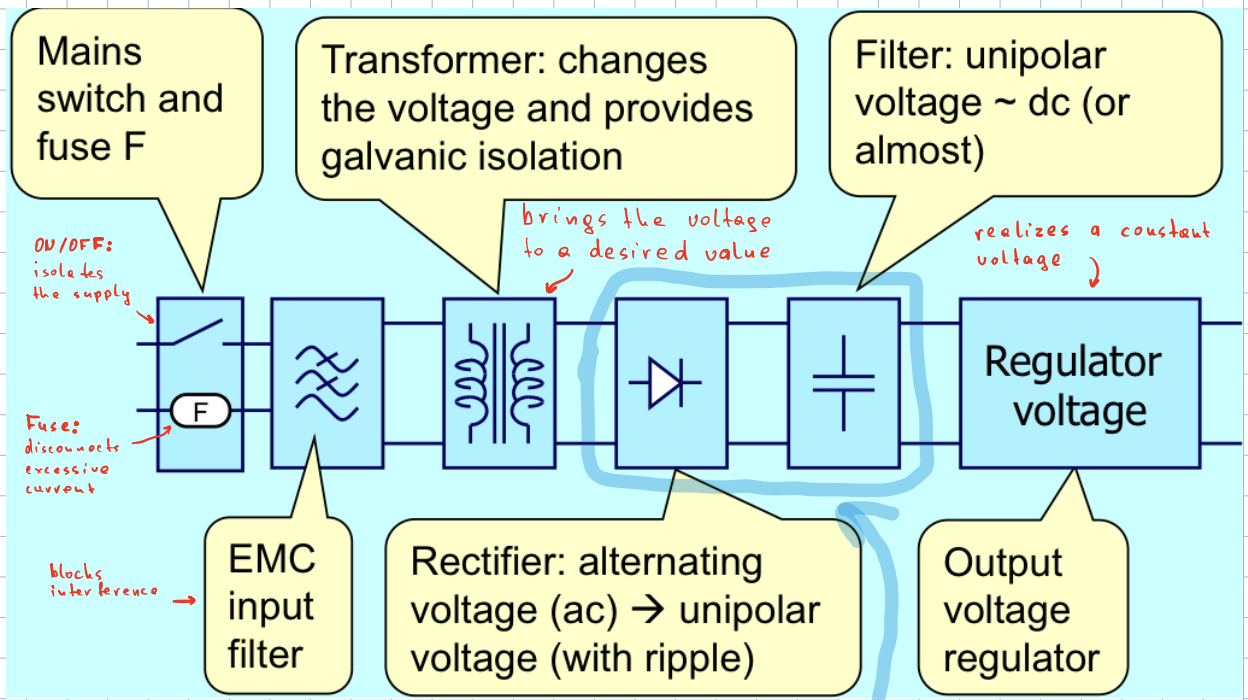


$T_J - T_A = P_D \cdot \theta_{JA}$

$T_J = T_A + P_D \cdot \theta_{JA} \leq T_{J\,MAX}$

Operating Limits

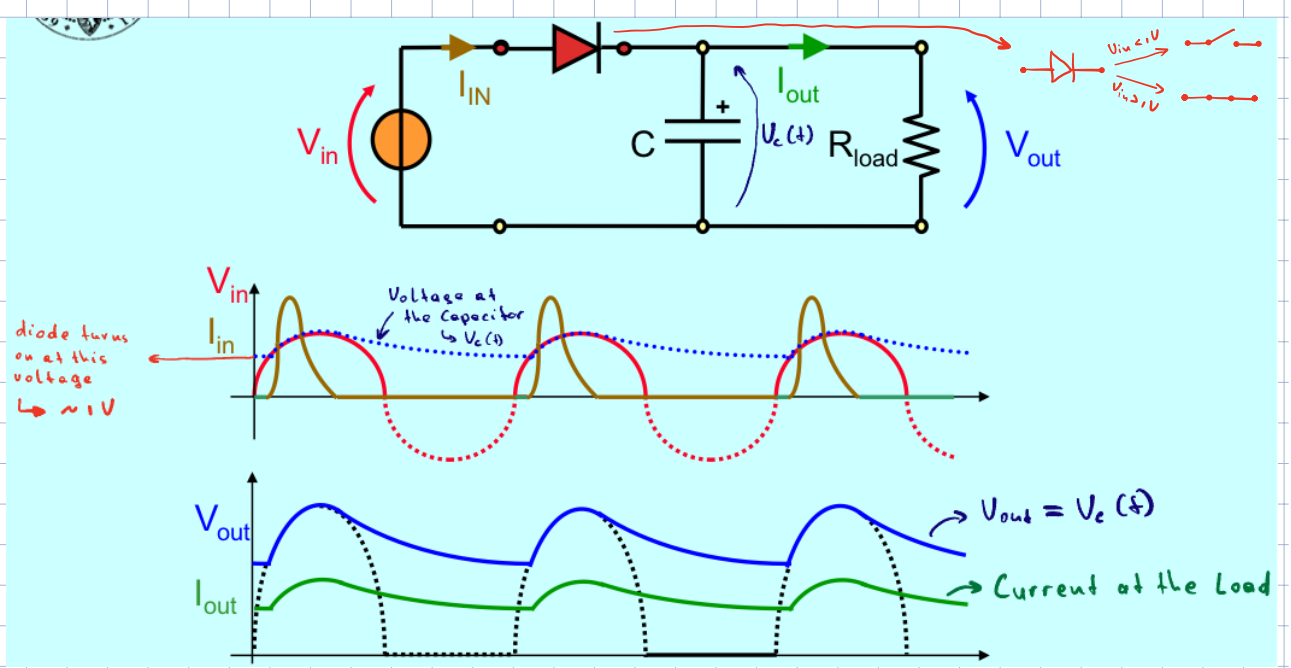
Block diagram of Power Supply Unit

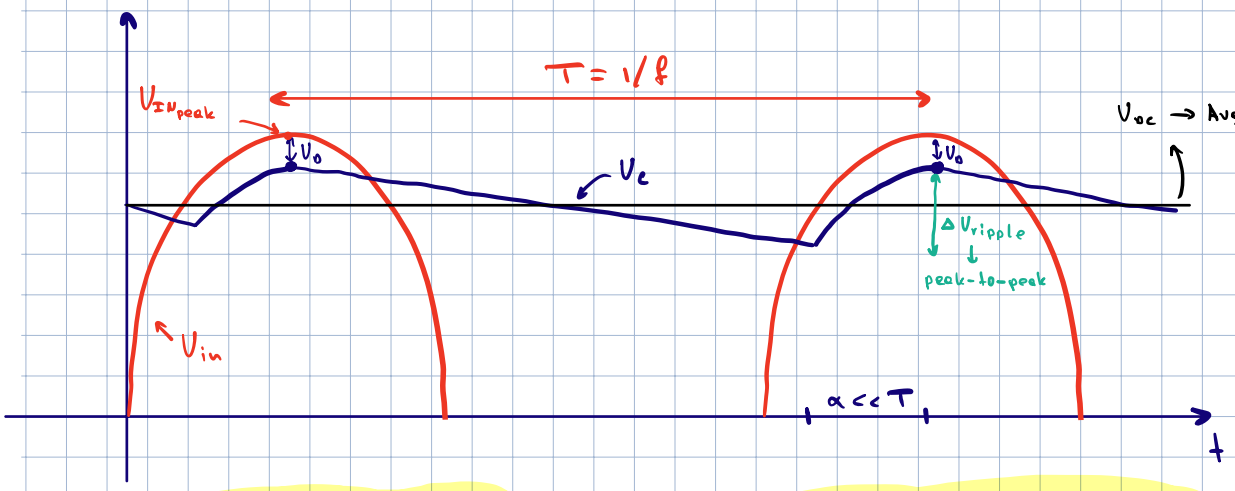


AC-DC conversion

Rectifiers

Half-wave rectifier

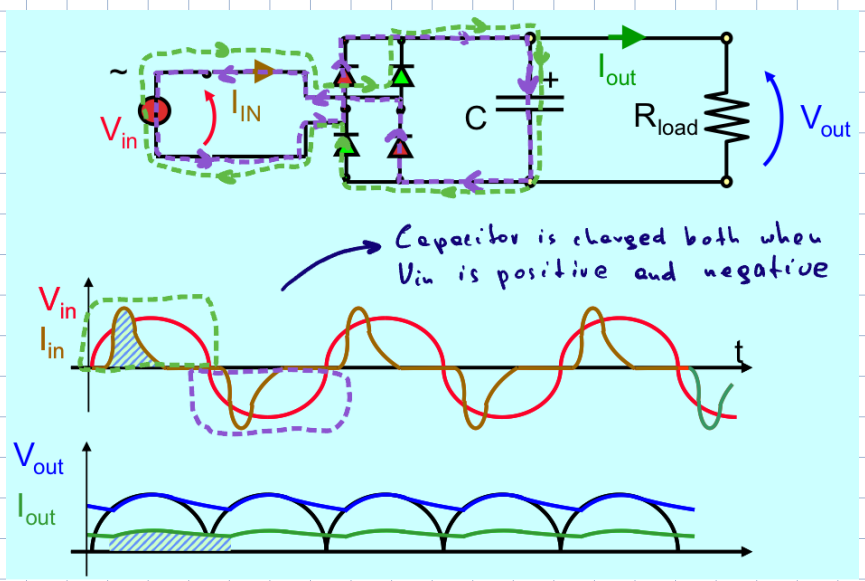




$$\Delta V_{ripple} = \frac{I_o}{C} \cdot T$$

$$V_{DC} = V_{IN_{peak}} - V_D - \frac{\Delta V_{ripp}}{2}$$

Full wave rectifier



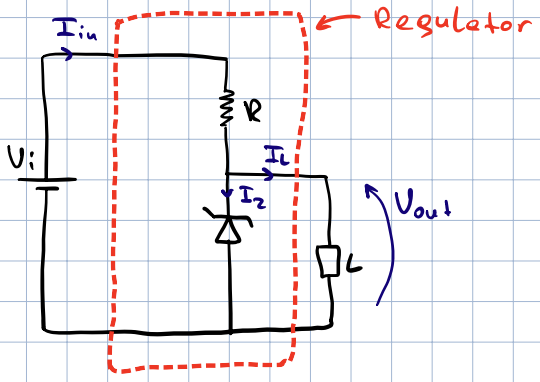
$$\Delta V_{ripp} = \frac{I_o}{C} \frac{T}{2}$$

$$V_{DC} = V_{IN_{peak}} - 2V_D - \frac{\Delta V_{ripp}}{2}$$

Voltage regulators

- Linear regulators
- Switching regulators

Linear Regulators → zener voltage (parallel)



$I_{in} = I_Z + I_L$

I_Z should be:

- higher than $I_{Z_{min}}$
- Lower than $I_{Z_{max}}$ (or $V_Z \cdot I_Z < P_{max}$)

efficiency:

$$\eta = \frac{P_{out}}{P_{in}} = \frac{V_{out} \cdot I_L}{V_{in} \cdot I_{in}} = \frac{V_{out}}{V_{in}} \frac{I_{in} - I_Z}{I_{in}} = \frac{V_{out}}{V_{in}}$$

$I_Z \simeq 0$

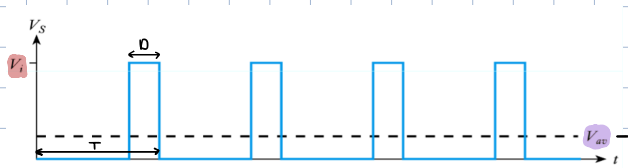
$$\eta \simeq \frac{V_{out}}{V_{in}} < 1$$ ← efficiency

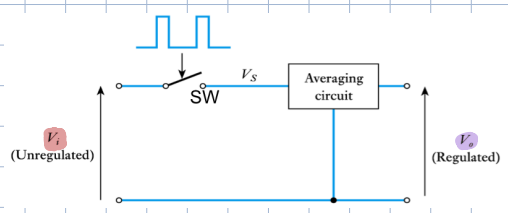
Problem → Low efficiency as $V_{out} < V_{in}$ always

↳ high power consumption

↳ Low frequency

Switching regulators

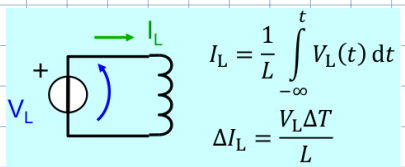




$$V_o = \frac{D T \cdot V_i}{T} = D V_i$$

↳ we can regulate V_o by modifying the Duty cycle (D)

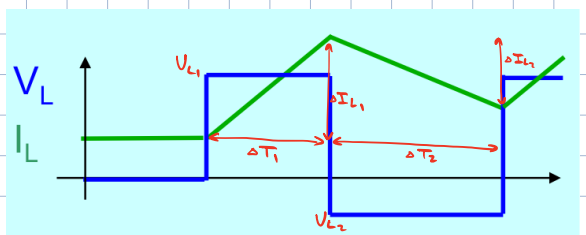
behaviour of an inductance



$$I_L = \frac{1}{L}\int_{-\infty}^{t} V_L(t)\,dt$$

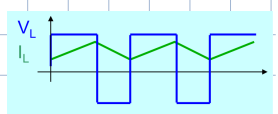
$$\Delta I_L = \frac{V_L \Delta T}{L}$$

if V_L is a square wave,



$$\Delta I_{L_1} = \frac{V_{L_1}}{L}\Delta T_1 \neq \Delta I_{L_2} = \frac{V_{L_2}}{L}\Delta T_2$$

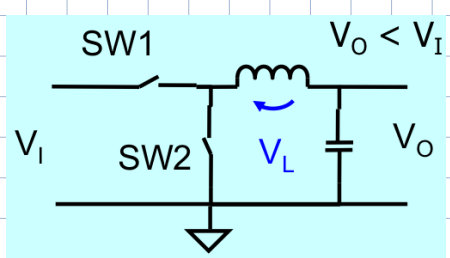
we need to make them equal, otherwise I_L diverges over time

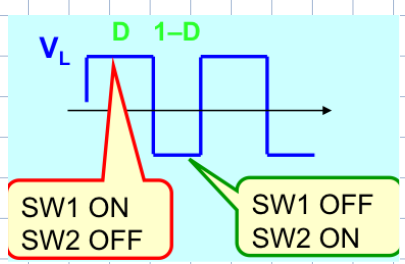


Switching regulator

- Buck reg.
- Boost reg.
- Buck-boost reg.
- Flyback

Buck regulator

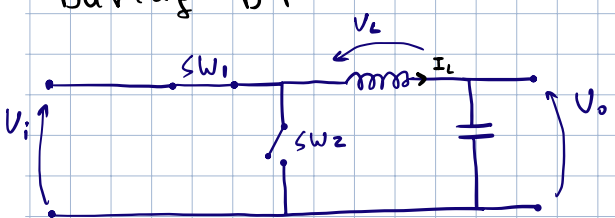




$$V_o = D\,V_i$$

We can control V_o by controlling D only

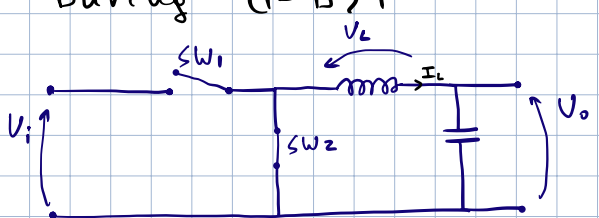
During DT



$$V_L(t) = V_i - V_o$$

$$\Delta I_L = \frac{1}{L}\int_{D\cdot T = T_{on}} V_L(t)\,dt = \frac{1}{L}(V_i - V_o)\,T_{on}$$

During (1 − D)T



$$V_L = -V_o$$

$$\Delta I_L = \frac{1}{L}\int_{(1-D)T = T_{off}} V_L(t)\,dt = -\frac{1}{L}V_o \cdot T_{off}$$

abs. value of ΔI_L above and below should be the same

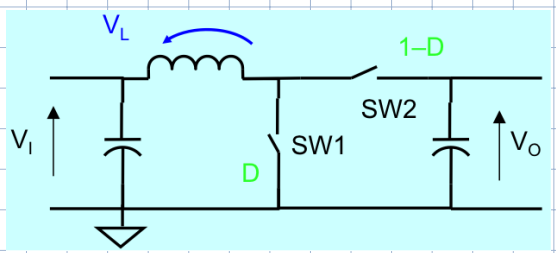
$$\left|\frac{1}{L}(V_i - V_o)\, D T\right| = \left|-\frac{1}{L} V_o \cdot (1-D) T\right| \rightarrow \boxed{V_o = D V_i}$$

efficiency:

$\eta = 1$ ← with ideal switches

$\eta < 1$ (close to 1: 0.8, 0.8), based of the switching freq.

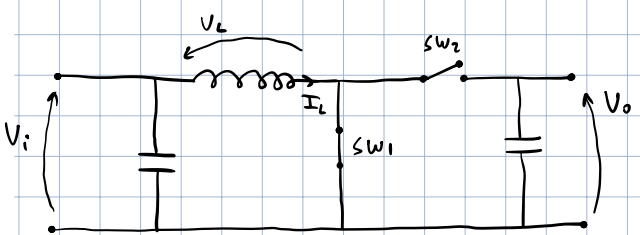
Boost regulator



$$V_{out} > V_{IN}$$

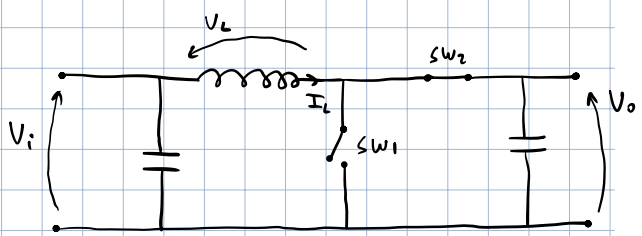
$$\boxed{\frac{V_O}{V_I} = \frac{1}{1-D}}$$

SW1 on



$$I_{L_1} = \frac{1}{L}\int_{0T} V_L\, dt = \frac{1}{L} V_i \cdot DT$$

SW2 on



$$I_{L_2} = \frac{1}{L}\int_{(1-D)T} V_L\, dt = \frac{1}{L}(V_i - V_o)(1-D)T$$

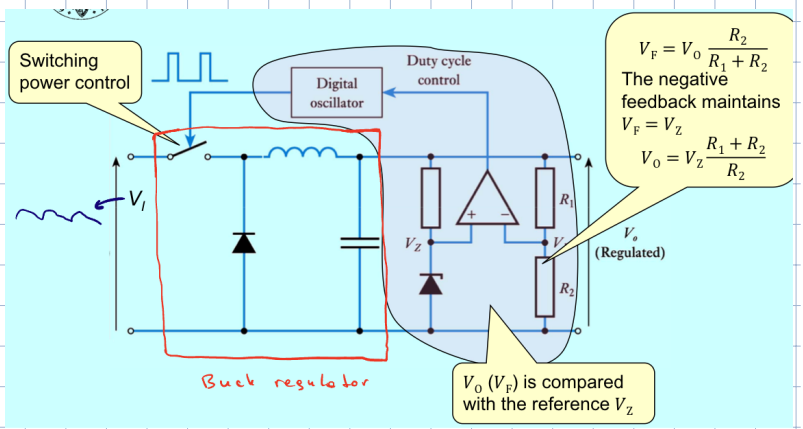
$$|I_{L_1}| = |I_{L_2}|$$

$$\left|\frac{1}{L} V_i D T\right| = \left|\frac{1}{L}(V_i - V_o)(1-D) T\right|$$

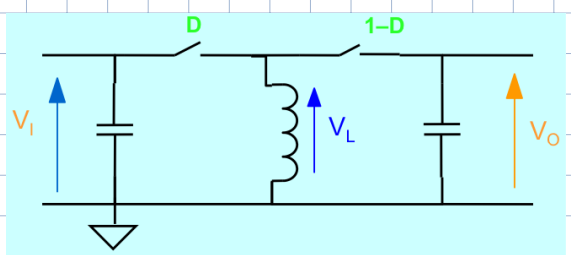
$$V_i D = V_o - V_o D - V_i + V_i D \rightarrow \boxed{\frac{V_o}{V_i} = \frac{1}{1-D}}$$

However, V_i from the rectifier is not constant

So, to regulate V_o, we change the Duty cycle following the variation of V_i



Buck-Boost regulator



- inverts the output voltage

$V_i > 0 \rightarrow V_o < 0$

$$\frac{V_o}{V_i} = -\frac{D}{1-D}$$

Flyback Power supply

Provides galvanic isolation

- Is a buck-boost regulator with a transformer instead of the inductance
- Control reaction is also isolated (optical coupling)