

RESEAU

• Réseau : ensemble de relations

- concret : physique, câbles
- abstrait

Communiquer : arriver à se joindre, se comprendre

Interface réseau : équipement pr passer du phy⁹ au log⁹

Identifiant : prop abstraite

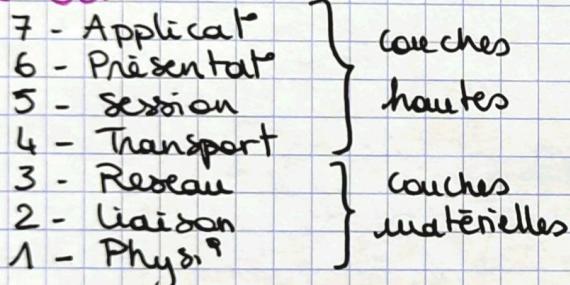
- identifier, nommer et distinguer les réseaux
- adresser : où est-il ?
- router : comment les joindre ? Info de topologie

Ex d' identificateur :

- Ethernet → MAC
- IP
- TCP → n° de port
- DNS → série de label précédé de la taille du label
- VFS

Tous n'ont pas les m^{es} fcts !

Modèle OSI



Mais concrètement on s'en faut du modèle OSI parce qu'on essaie même plus de faire coller les protocoles au modèle OSI

Réseau logique

Historique → cf cours de géopolitique de l'Internet

Concept des protocoles au début : utiliser le moins de bits possibles
Pb : comment s'échanger les infos? → réunion pour décider
→ 1^{er} RFC de John Postel

Bittorrent : pas une RFC. Bcp de souci : bencode, ...

xxx . x x x . x . x x / yy yy : nbr de bit pr le réseau

le reste permet d'identifier les machines sur le réseau

Quand réseau src associe paquet + IP src + IP dst + MAC src + MAC dst
ARP → envoie msg avec H MAC dst à 1. → on récupère MAC dst grâce à la réponse

Puis envoie : soit machine si m^{es} réseau

soit switch sinon → routage du paquet w/ buffer

Quand réseau dst : si bonne addr MAC → send to OS
sinon → drop paquet

Si pas co à réseau où se trouve dst → machine passerelle.
On envoie les paquets à passerelle avec MAC de passerelle au lieu de dst.

Chaque passerelle forward le paquet de src jusqu'à dst

plus d'IPv4. → Plage allouée au ministère de la Défense USA
énorme, pas utilisées... et pas accessible d'Internet.
→ utilisée en interne.

(NAT) → Transforme paquet par passerelle, comme si c'était elle qui était src. + table d'état pour retrouver (FTP, TCP / UDP)

FTP : connexion | de contrôle
| de donnée

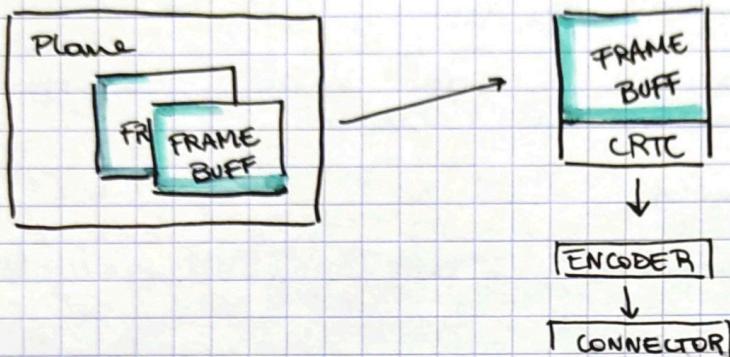
NAT : network addr transformation

GRAPHIC STACK

OPENGL

On ouvre 1 fenêtre. Clear, créer les vertex, and, flush
 → on affiche des choses!

On dessine ds 1 space mem → Frame buffer
 On peut superposer les frame buffer



Partie 2D: Serveur X → connexion, recevoir cmd
 → lib xcb (wrapper)
 Je veux dessiner à l'écran, fais ce que tu veux mais
 soit au courant.

Partie 3D: Mesa ⇒ libGL (symboles, interface pr brancher)

↓
 State tracker (machine à état
 retiens vertice, couleurs, etc)
 ↓
 DRM Driver (envoyer au GPU)

DRM: (abstract du GPU) → alloc
 HW (GPU)

DRM: Generic API
 ↘ KMS ↗ TTM & GEM ↗ PRIME

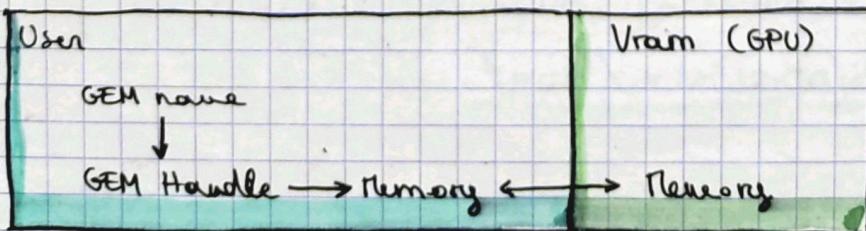
Aut chaque appli gérait l'affichage → conflits ds ts les sens
 Tlfn → Kernel.

KMS:

TTM & GEM : alloc mem et shared à userland
 tis lourd & complexe ↗ mem unifiée
 Interface GEM avec TTM dernière

PRIME: pour gérer le true avec 2 GPU
 ↗ partage la mémoire

GEM:



Cmd envoyée au GPU : alloc, display, ...

"Je veux créer un programme, voici mon shader, exécute-le avec ces param"

Sur Windows : à peu près la même chose.

Block 2D aussi pour les imprimantes.

On parle pas directement au device ! On passe par le driver Windows de lui envoyer

VirtIO

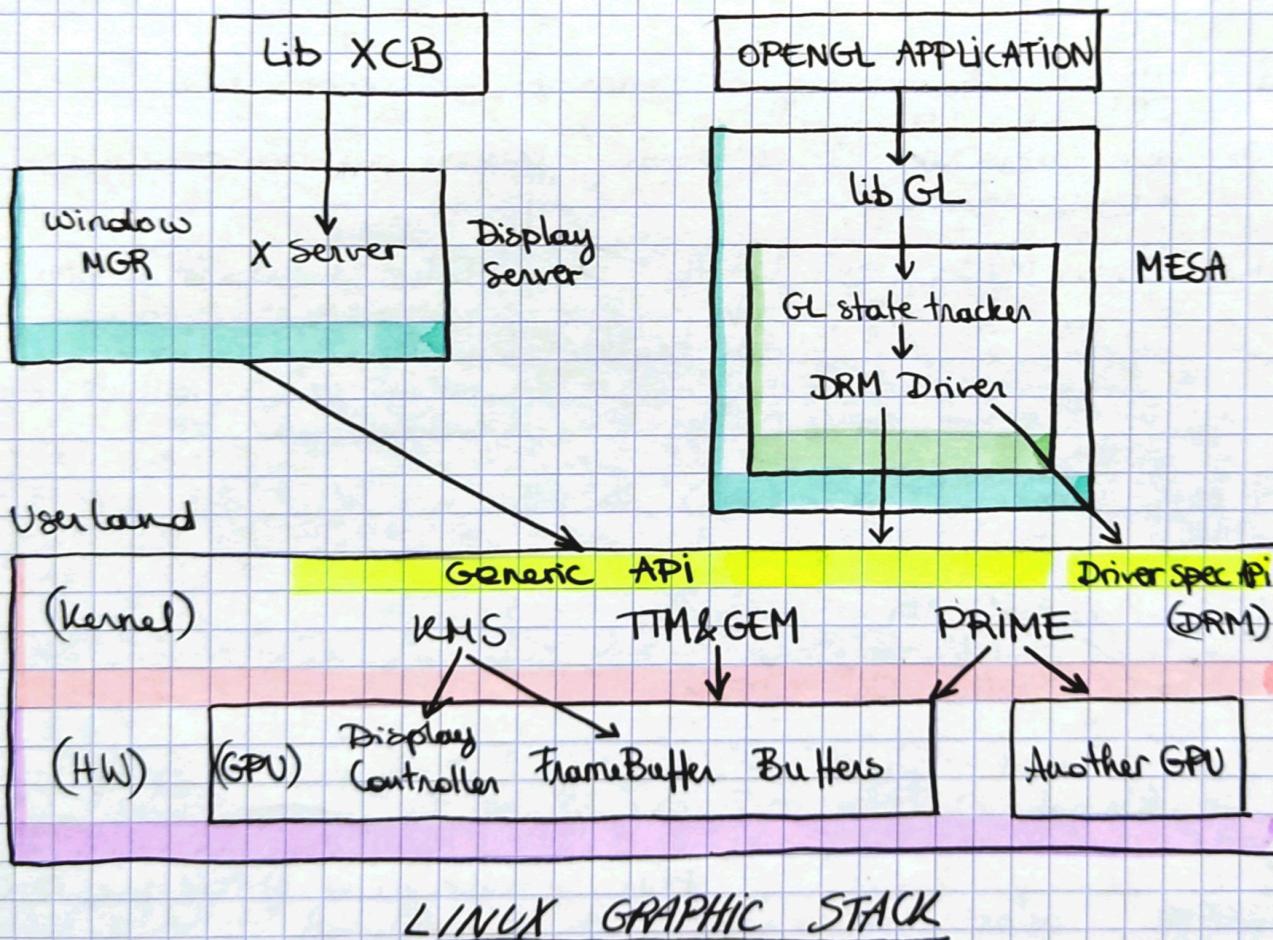
FIFO pour envoyer cmd

VirtIO → direct access à la FIFO

Sur VM → on veut pas que le guest écrive n'importe où il faut passer par l'HOST.

VirGL : écrit de l'OpenGL

Avantage : si support OpenGL, on peut lancer VM avec acceleration 3D



Pour les autres schémas et plus d'explications :

www.studiopixl.com/blog/

LINUX KERNEL EXPLOIT

static linked ELF. Ring 0

Vuln commun en userland. → C'est du C

↳ stack overflow, null ptr deref, stabs exploit, race condition, ...
⇒ syscall vulnerabilities
native ou exposées par device

★ NULL ptr deref

→ page fault, segfault
si kernel → triple fault, panic (dump tout & reboot)

Rip 0 et exec code dessus

↳ map page 0 avec MAP_FIXED
ou copie du code ici
ou trigger le NULL ptr deref

user peut pas accéder à kernel page
mais kernel peut accéder à user page!

On veut passer Root et exec en kernel

Kernel symbols : /proc/kallsyms ou System.map (^{pas tous} à _{static})
(dynamic)

prepare_kernel_cred arg = 0 → init tout à root
commit_creds assign struct à process
→ process en root!

Thitigat : • munmap_min_addr min addr pr map
• smep smap protect x86
↳ empêche : idem en
d'accéder à page user : accès RW
• PAX, KERNEEXEC, UDEREF ~ pareil que smep smap.

★ Stack buffer overflow

écrire plus loin que le buffer
→ overflow sur stack et recréer addr de retour

- recréer addr de retour sur pile
-
- root
- ret
- fucked up stack
-
- panic

→ On doit retourner en userland proprement : itet

iret : fausse frame en settant des choses bien

- eip sur shell
- CS SS
- stack ptr
- efflags

- retrieve CS DS ESP
- crack iret frame
- trigger vuln
- exec privilege escalation
- take iret on stack
- iret to payload
- execve
- root shell !

Tutigati : smep smap

Bypass:

- ~ Full ROP
- ~ désactiver SMEP SMAP en ROP des CR4
- ~ kernel bytecode

Conclusion: c'est une très petite intro
Il ya vraiment plein de trucs assez divers

CRYPTO

: Crypto asymétrique :

crypto symétrique : comme deux clés identiques pour ouvrir et fermer la porte.

Pb actuel : crypto asym' trop lent pr chiffrer gros trucs.

Utilise nbres très grand (CTF : Python - trop grand pr C)

Ensemble $\mathbb{Z}/n\mathbb{Z} \simeq \mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$
n 1° pas RSA
n composé, sûrement RSA
ensemble utilisé pour courbes elliptiques.

Tms de clé RSA neg. Tjs ≥ 0

Bin opé ° : addit° + , multi° * , ou autre

Equ° qu'on peut résoudre ou non

2 pb princ. en crypto : • factorisat°
• discrete logarithm

factorisat° :

$n = x^2 - y^2 = (x+y)(x-y)$ si $x-y \neq 1 \rightarrow$ facto
marche sur une plage précise

$n = pq$ si $|p-q| < n^{1/4}$

$\Rightarrow x^2 - n = +y^2$ en 2 itérat° algo cassé ?
 $x = \lfloor \sqrt{n} \rfloor$ $x^2 - n \approx y^2$

~ 10aine d'algo

discrete logarithm

$y, g : \mathbb{Z}_n^*$ $y = g^\alpha$ $\log y = \alpha \log g \text{ TR}$

Pb qui change en fonction de la modularité des clés.

$y^2 = a \pmod p$ $\xrightarrow{\text{premier trivial}}$

!premier pas d'algo!

Algo de base : ne pas recoder ce qui a déjà été fait
• Python 2 et 3 • Mathematica • ...

le + important en crypto asymétrique : Fermat's little theorem.

Img - permutat° au hasard - On retombe toujours sur l'image au bout d'un moment.

Struct de base :

- Groupe
- Ring
- Field (corps)

} tous les algos de crypto utilise au moins 1 des 3

grp : Closure - Existence of Identity - Associativity - Inverse -
Commutativity ↳ primalité des

Ring : Closure - Associativity - neutral elmt - Distributivity

Fields : Closure - Associativity - Commutativity - Identity elmt - Inverse
- Distributivity.



Rpz non ensembleliste

Matrice : marche pas en crypto :)

Calcul modulaire : ex & prop in slides.
arithmetic

Thm de Bezout \rightarrow calcul de clé RSA

Courbes elliptiques mi-XIX^{es}.

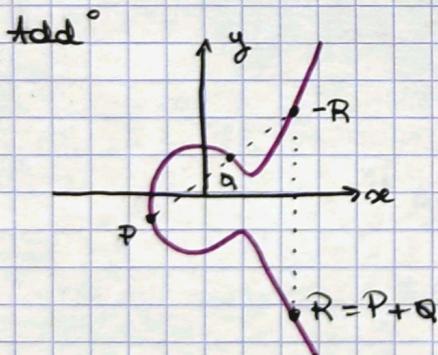
va sûrement bientôt prendre le dessus en crypto.

- Elmt neutre O point à l'infini
- On n'aime pas les ensembles infinis \Rightarrow mod p.
- ~~Y~~ Tous les pts vérifiant $y^2 = x^3 + ax + b \text{ mod } p$
et O pt à l'infini.

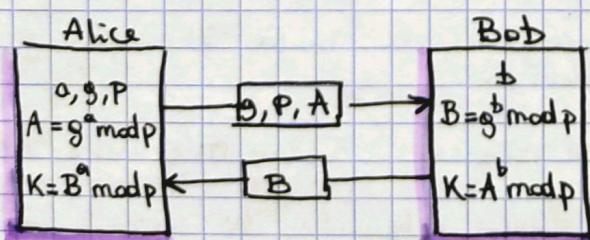
Pb \Rightarrow ne se calcule pas facilement :/

\rightarrow nbr d'éléments dépend de p mais très difficile à calculer : choisit donc courbe déjà calculées ...

\hookrightarrow courbes safe et unsafe (safe curves. cr. yp. to /)



Diffie - Hellman



a} random, privés

b} g, p, A, B publics

Version naïve

CRYPTO 2.

Algo construct^o public key RSA:

P & Q prime

$$N = PQ$$

$$ED = 1 \bmod (P-1)(Q-1)$$

$$C = M^E \bmod N$$

$$M = C^D \bmod N$$

Algo naïf / std^t

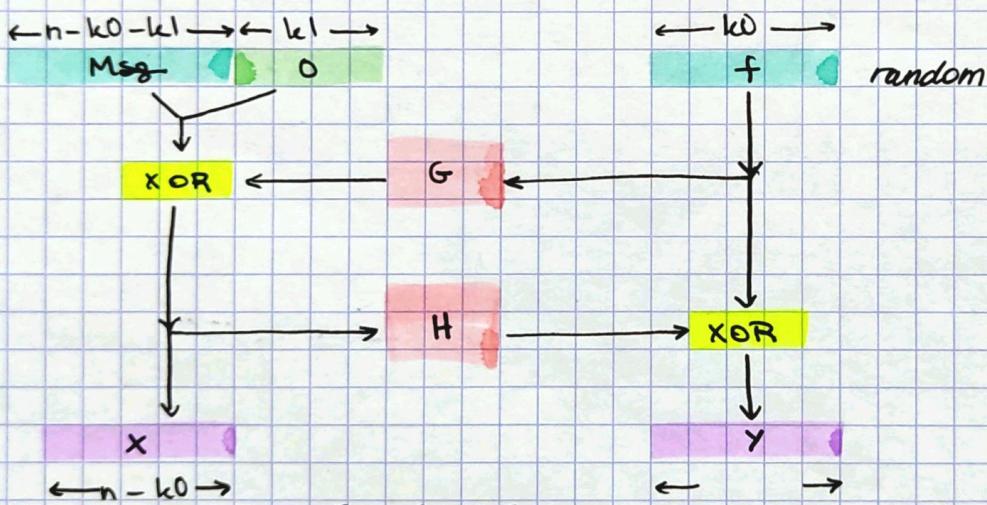
- 1: 2 \neq large random prime nb p & q
- 2: $n = p \times q$ modulo
- 3: Euler totient $\varphi(n) = (p-1)(q-1)$
- 4: $1 < e < \varphi(n)$ e coprime to $\varphi(n)$
- 5:

- Chiffrage: $c = m^e \bmod n$
- Déchiffrage: $m = c^d \bmod n$

RSA : sent aussi à signer

$$\text{sign: } S(M) = M^d \bmod n$$

$$\text{verify: } S(M)^e = (M^d)^e \bmod n = M^{ed} \bmod n = M \bmod n$$



Pre-Snowden : quelques personnes qui disent qu'il faut avoir une vue panoramique de la crypto

- Pb:
- * clé qui vieillit
 - * Algo qui vieillit
 - * Mal codé
 - * Erreur HW
 - * Protocole insecure. use
 - * Processor ds device non sécurisé
 - * Backdoor ds Algo, proc, key

Attack

- * Cycling attack (Morris & Simmonds)
- * Common modulus attack (Simmons)
- * Broadcast attack (Hastad)
- * Small public exponent attack (based on Lattice Attacks)

CONCOLIC

Concrete exec + Symbolic exec



exec sur
machine
sur CPU

bin → emulat^t
en analysant

lent mais on voit tout ce qui se passe

Rpz abstraite → arbre de contrainte

Pratique en vers d'énorme pgm
emulat^t de toutes les branches.

pr aller au résultat attendu → donne les les contraintes

→ angr (+ d'autres équivalent) : uniquement symbolique

analyse concrete pr aller + rapidement qd ya que des maths
on lance sur CPU et on récupère entrée et sortie

ex ds slides de la LSE Week 2016

Pb : modèle Kernel , ptr sur fct , blocks pas complets, ...

Surtout utiles pr CTF

FPGA

TTL : consomme ++ que CMOS
 → préférer CMOS

Circuit combinatoire :

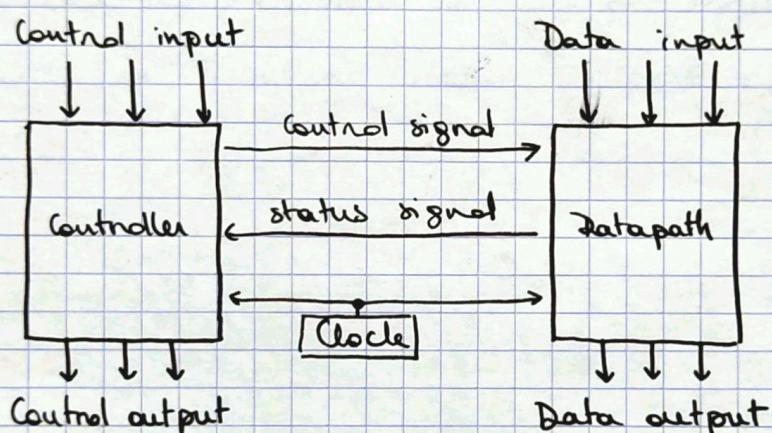
fct logique avec entrée et sortie
 → table de vérité et équation logique ⇒ circuit

D_o not D et D ou D xor

Stock State à un moment ⇒ machine à état ou registres

RTL :

2 chemin control path data path
 opé données



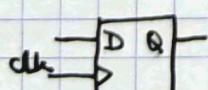
→ Séparer data et control + interface entre les deux

Refléchir sous forme de flow de données

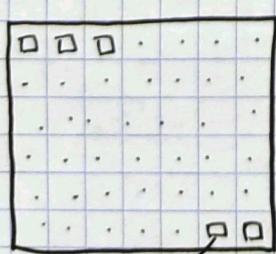
Verilog : reg conserve un état
 wire var fil ~~est tracé~~ → ne conserve rien

si pls bit type [pd fort : pd faible] var sinon type var

registre : bascule \square



entrée data \square sortie $Q = D$
 clock maj à chaque front de clock



\square signaux d'entrée tables de vérité

FPGA

2 lang : VHDL et Verilog → + de liberté, + utilisée des outils libres

↳ assert dans le code
syntax très lourde

Tools : Icarus Verilog → simule design
GTK Wave → voir les traces sorties par]

Verilator → simule design et permet d'instrumenter avec du C++
Altera Quartus II → à cracké, fait des tool chain (?)

logique synchrone : se base sur signal d'horloge

2 types d'assignat° :

bloquante = valeur curr (se font l'une après l'autre)
non bloquante <= valeur d'aut (se font en même temps)

$b = a$
 $c = b \rightarrow c = a$

$b \leftarrow a$
 $c \leftarrow b \rightarrow c$ vaut ancienne valeur de b

Dans bloc synchrone → que des assignat° non bloquantes.

Directives : \$directive dispo uniquement en simulat°

PCB

PCB : printed circuit board

- ★ Reconnaissance et tester
- ★ Schéma
- ★
- ★ Souder

Avant de se lancer → prototype Important pr pas perdre tps et \$

Breadboard pr prototype breakout board pr tester

≠ types de composants:

- DIP (utilisable sur breadboard)

Small outline (SOIC) → jouable au fer à souder

Thin shrink Small outline package → pate de bordure

BGA (pin en dessous de la puce) → four, impossible à la main
à souder

Choisir et acheter:

- tannell (pro)
- mouser (Europe :)) → doc ++
- digikey (USA :/)

! lire bcp datasheet qd draft PCB !

→ c'est là qu'on passe le + de tps.

logiciel :

- | | |
|---------------------------------------|---------------------------------|
| • kicad (open source) | → interface graphique, + fourni |
| • geda | → plein de petites étapes |
| • altium (simulateur, vrai pb ou pas) | (commercial) |
| • eagle (plus d'intérêt) | |
| • orcad | |

1 - Sélectionner composants en fonction de fonctionnalité

2 - Footprint (empreinte de la puce)

Q4 ↳ à partir de schéma, on place les composants

But : faire les + courts possibles, composants les + proches possible
H en restant ds les limites de la PCB

Risques: chauffe, parasite, perte de précision

Auto routing : Bof bof

Plz couches pr mettre les pistes

le + souvent 2 pistes (echo vero)

des que trop gros, couches cachées en + (+ de calcul, ...)

Pb princ: traces + courtes possibles, agencement logique, dissiper chaleur

+ de couches → + complexe

Plan de masse : grosse plaque dédiée

- logique : à niveau, haut et bas, 1 et 0, vrai et faux (signal carié)
 - analogique : sinusoidale
- Qd finit de désigner, constructeur
→ ITEAD

Etape suivante du prototypage : check si électriquement correct → ng-spice
(contraintes physiques)

PCB manufacturer :

- ITEAD
- OSHT Park (\$ x 10, + cher)
-

BOM : build of material → excel avec ref, prix, quantité, ...
(à faire tôt au début)

↳ prendre en compte l'intensité du circuit

Une fois que l'on a PCB et composants

⇒ Soudé :)

flux à souder (meilleur répartit° de l'étain) (à conserver au frigo)
pâte à souder = flux + étain

Soudé c'est dur avec des outils de menuiserie

- evertools
- hakko day
- dangerous prototypes

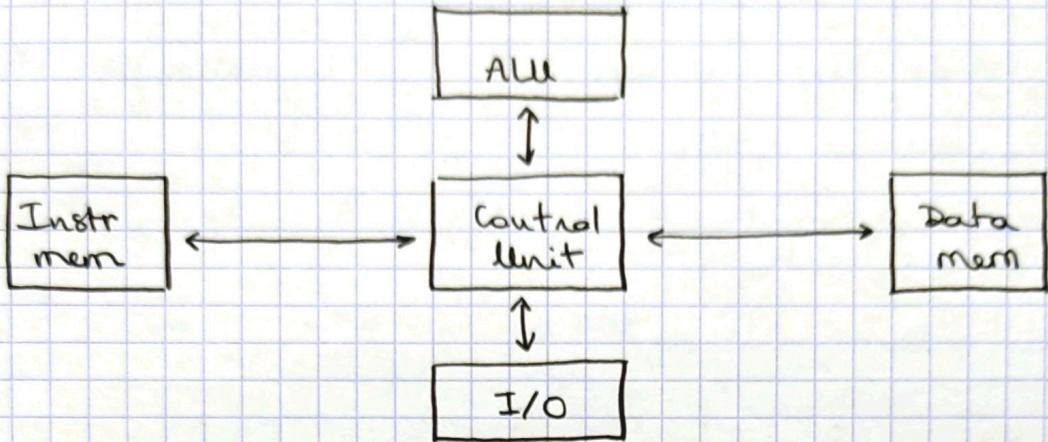
Bus pirate : interface usb ↔ protocole de board

Bus magique : pareil que bus pirate mais en + pert.

AVR

- Archi de microcontroller

au fur et à mesure des années, transistors + petits et - consu



Flash : program et data perm
Control unit : décode instr ()

≠ Reg 32 de 8 bytes → interact avec ALU

Program counter : où on est ds programme

EEPROM : var de config + long que dyn ? RO.
bouge pas pendant exec.

Reg intéressant : les 6 derniers : ptr → addr R16 - R31
+ R0 et R1
↓ ↳ tjs 0
tmp
pas besoin de svg.

1 tic = 1 instr → pipeline : fetch execute (slide 6-7-8)

tab d' interrupt : par chaque 1ptr sur fct sur lequel jump
il y a aussi interrupt sur pin

sei set enable interrupt
cli interrupt disable
reti revenir d'interrupt

pgm mem accessible uniquement pr pgm
EEPROM → instr part pour y accéder (slide 18)

Reg spé pr config microcontroller et accès
SRAM : dyn mem

jmp [Boot loader : code pr reprogrammer puce
Reset vector]

Fuses bits : zone mem def par machine qui pgm la puce
→ config + perm.

oscillateur ds la puce \rightarrow horloge interne
mais on peut avoir horloge externe (cristal qui vibre)

condensateur \rightarrow impact sur oscillat° du cristal

gest° d'alim°

ADC Noise reduc° (grasse intra, mode + précis)
Idle

Reset logic (slide 26) \rightarrow reset

Watchdog timer (slide 27)

MCU : micro controller unit

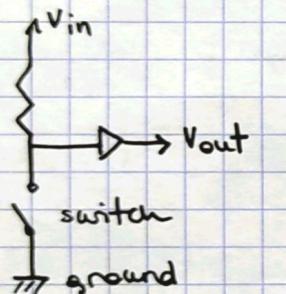
 multiplexer : plein d'entrées, signaux de control, 1 sortie
~table de vérité

$\rightarrow \Sigma$ stock de valeur (ds reg.)

Tri-state logic : 3 états. vrai, faux, flottant (ouvert)



Solut° à l'état flottant : Pullup resistor



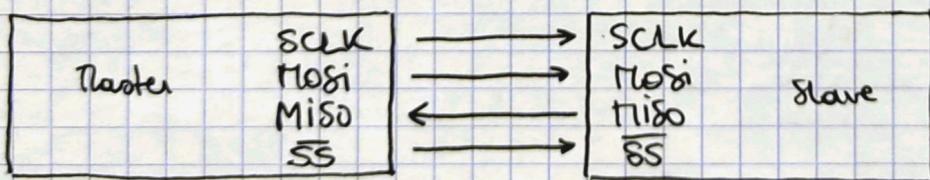
ouvert : état haut

fermé : pent à la masse

(cf Elec° prépa)

Péroph : communiquer ac protocole série.

SPI 4 pins comm ac flash ou EEPROM
1 master 1 slave
init request exec request



Frame \rightarrow slide 37

P parité : correct d'erreur (exo réseau S5)

IDLE : qd il ne passe rien.

TWI (I2C) pin intéressantes

master met addr de device ds frame
si correspond \rightarrow slave communique

SCL master slave
SDA data

Frame slide 39

\Rightarrow synchro
Signal d'horloge
Master out slave in
Master in slave out
slave select
↳ select slave

AVR 2.

Convertisseur analogique \leftrightarrow numérique

Protocole PWM {pulse width modulation}

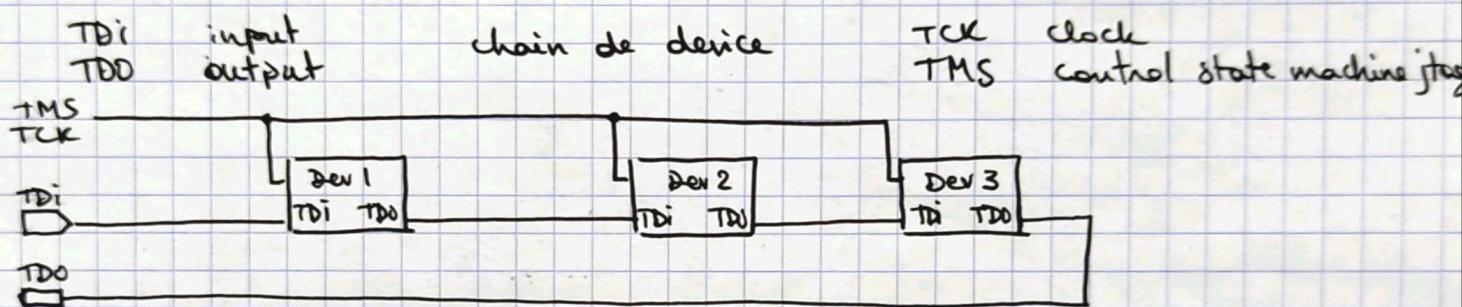
interval entre 2 pics fait monter ou descendre signal (slide 2)

2 autres périph : LCD controller USB controller (Talk à voir)
 → super complexe

Calling convention slide 4.5

// port : protocol quasi plus utilisé (plus que les pointeurs)

JTag Chain : au départ \rightarrow vérif de pin
 étendue pour le debugging ac bp, rw mem
 \rightarrow incontournable en debug et reverse



Debug wire idem mais 1 seul fil \rightarrow + complexe

JTagulator \rightarrow trouve le(s) pin(s) jtag
 sans casser la board !