

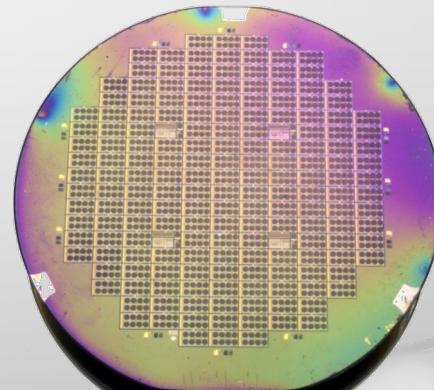
CONCEPTION DES SYSTÈMES NUMÉRIQUES

2A PARCOURS
SYSTÈMES AUTONOMES
SYSTÈMES ET LOGICIELS EMBARQUÉS



Conception des systèmes numériques

- Introduction
- Algèbre de Boole et simplification logique
- Modélisation: introduction à la modélisation VHDL
- Systèmes combinatoires
- Systèmes séquentiels
- Machine à états finis



Programme théorique : CM

6 heures de cours, 3 séances de 2 heures

- CM1 :
 - classification des cibles d'intégration : comparaison des solutions programmées et câblées. Du cahier des charges aux spécifications jusqu'à la conception : analyses fonctionnelles
 - algèbre de Boole rappels
- CM2 : modélisation des composants logiques, simulation logique et modélisation VHDL
 - méthodes de description des systèmes combinatoires, séquentiels
 - description hiérarchisée de systèmes complexes
- CM3 :
 - conception et mise en équation de systèmes combinatoires et modélisation VHDL
 - conception et mise en équation de systèmes séquentiels jusqu'à la machine à états finis et modélisation VHD

Programme travaux dirigés

4 heures de TD, 2 séances de 2 heures

- TD1 :
 - algèbre de Boole rappels
 - Code VHDL
- TD2 :
 - conception et mise en équation de systèmes combinatoires
 - conception et mise en équation de systèmes séquentiels

Programme travaux pratiques

16 heures de TP, 4 séances de 4 heures (Systèmes autonomes)

20 heures de TP, 5 séances de 4 heures (Systèmes et logiciels embarqués)

- Introduction à la synthèse logique, méthodologie de conception d'un FPGA : Prise en main du logiciel Quartus, déclaration d'un projet, description graphique, déclaration de la cible matérielle FPGA (carte DE1-SoC), analyse fonctionnelle et temporelle du processus décrit
- Présentation et étude **d'un simulateur professionnel** (MODELSIM) et la notion de test : écriture d'un programme de test (benchmark) Utilisation d'un simulateur pour mesurer la «couverture de code»
- Description VHDL de solutions combinatoires puis câblages sur cible FPGA, analyses fonctionnelles et temporelles couverture de code : l'additionneur 4 bits
- Description VHDL de solutions séquentielles puis câblages de cible FPGA, analyses fonctionnelles et temporelles couverture de code : horloges, compteurs

Introduction générale

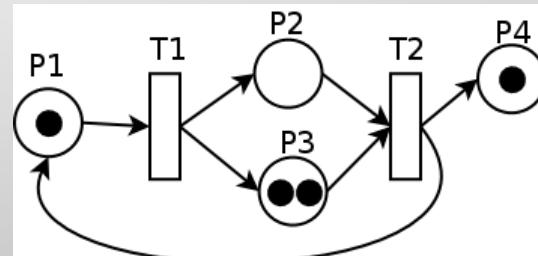
- ▶ L'électronique numérique n'est pas une modélisation de phénomènes naturels comme la physique, la chimie ou la biologie...
- ▶ C'est une technique élaborée par l'homme pour se doter d'outils dédiés à diverses fonctions.
- ▶ Elle implique un certain nombre de concepts fondamentaux dans le but de synthétiser systématiquement des systèmes.

Introduction générale

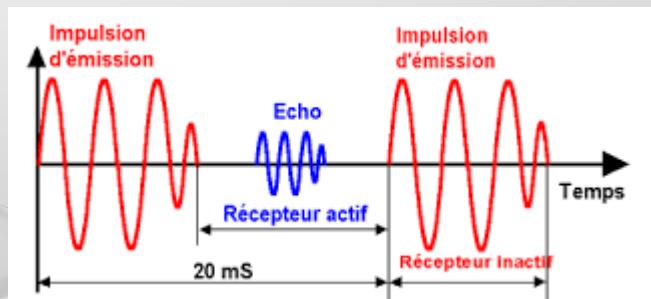
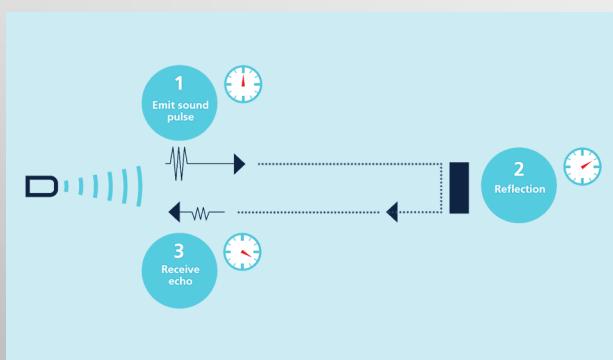
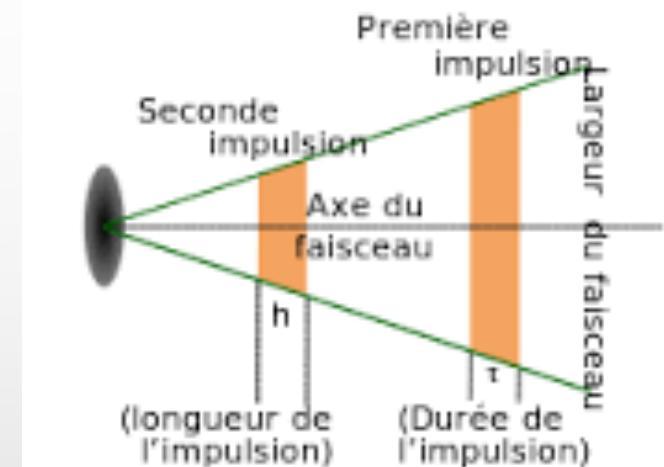
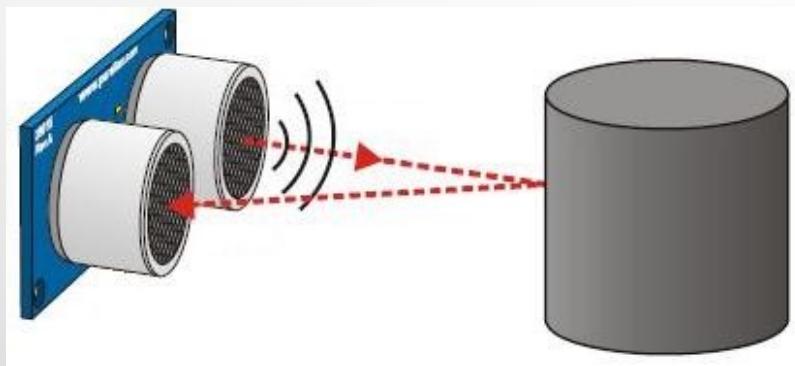
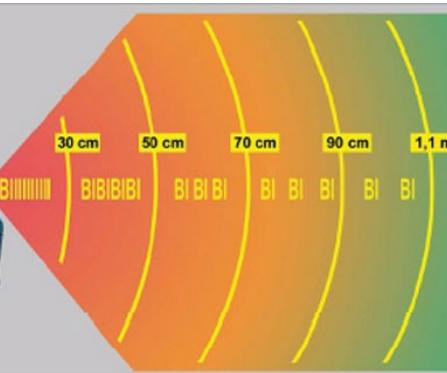
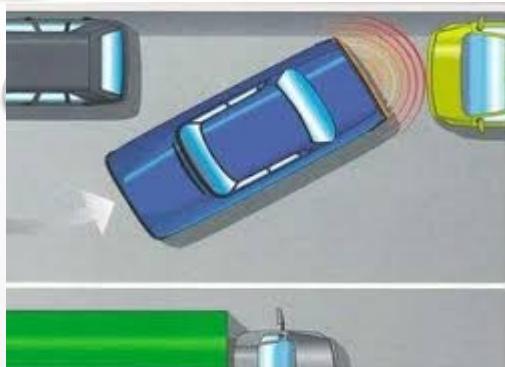
- L'**électronique numérique** regroupe les systèmes électroniques fonctionnant sur la base **d'états électriques** précis dont le nombre et les valeurs sont fixés dès leur conception
→ À chaque état correspond une valeur numérique.
- La prédétermination de ces états électriques permet de disposer de systèmes qui se comportent de manière stable et fiable.
- Ce type d'électronique est opposé à l'**électronique analogique** qui traite des systèmes électroniques opérant sur des grandeurs (tension, courant, charge) continues.

Introduction générale

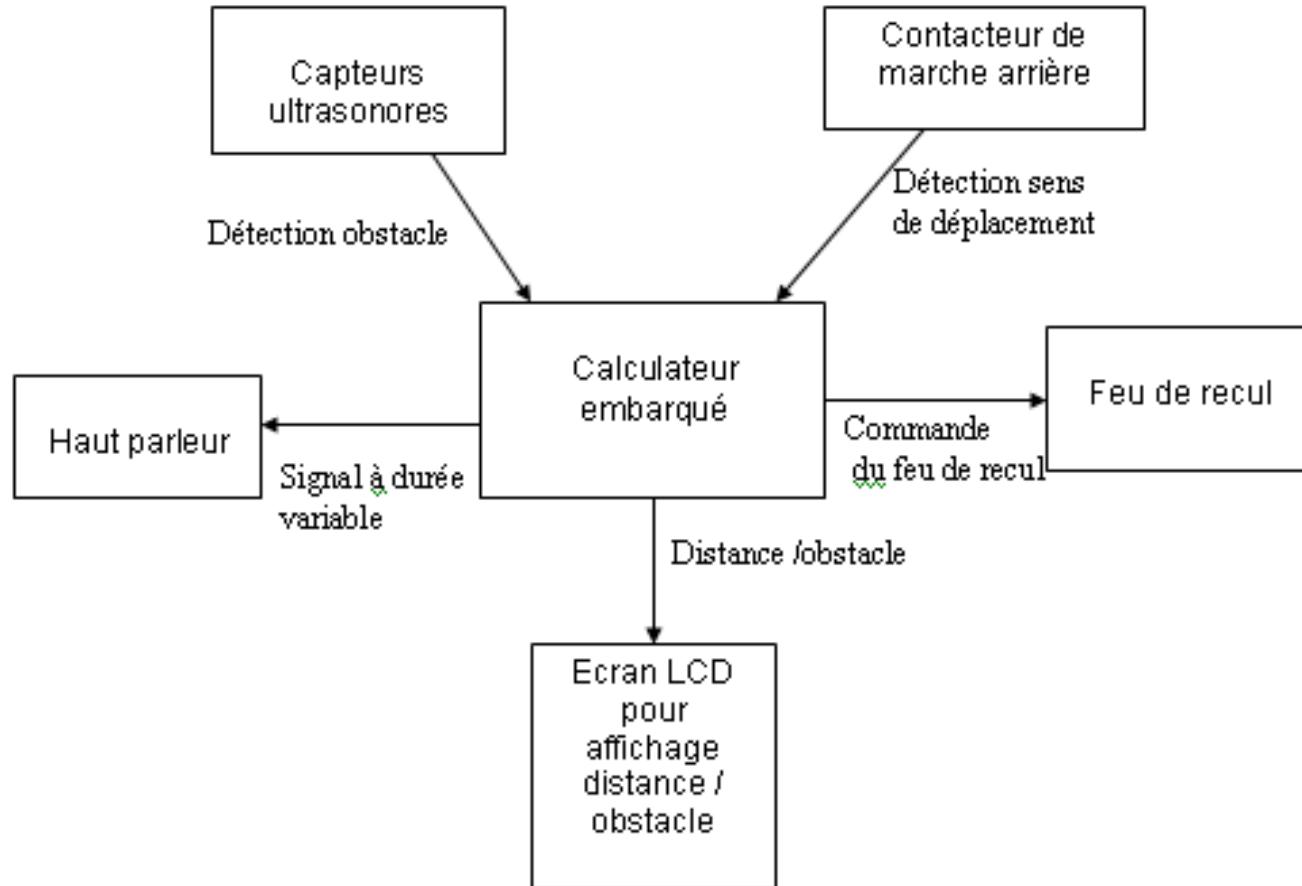
- En réduisant le nombre d'états distincts que peut prendre un signal
→ moins sensibles aux perturbations.
- Les états sont notés **faux / vrai**
- Quantification des erreurs : code correcteur d'erreur
- Possibilité de mise en œuvre d'une arithmétique basée sur la logique
- Modélisation systèmes à évènements discrets SED par exemple réseaux de pétri



Etude : radar de recul

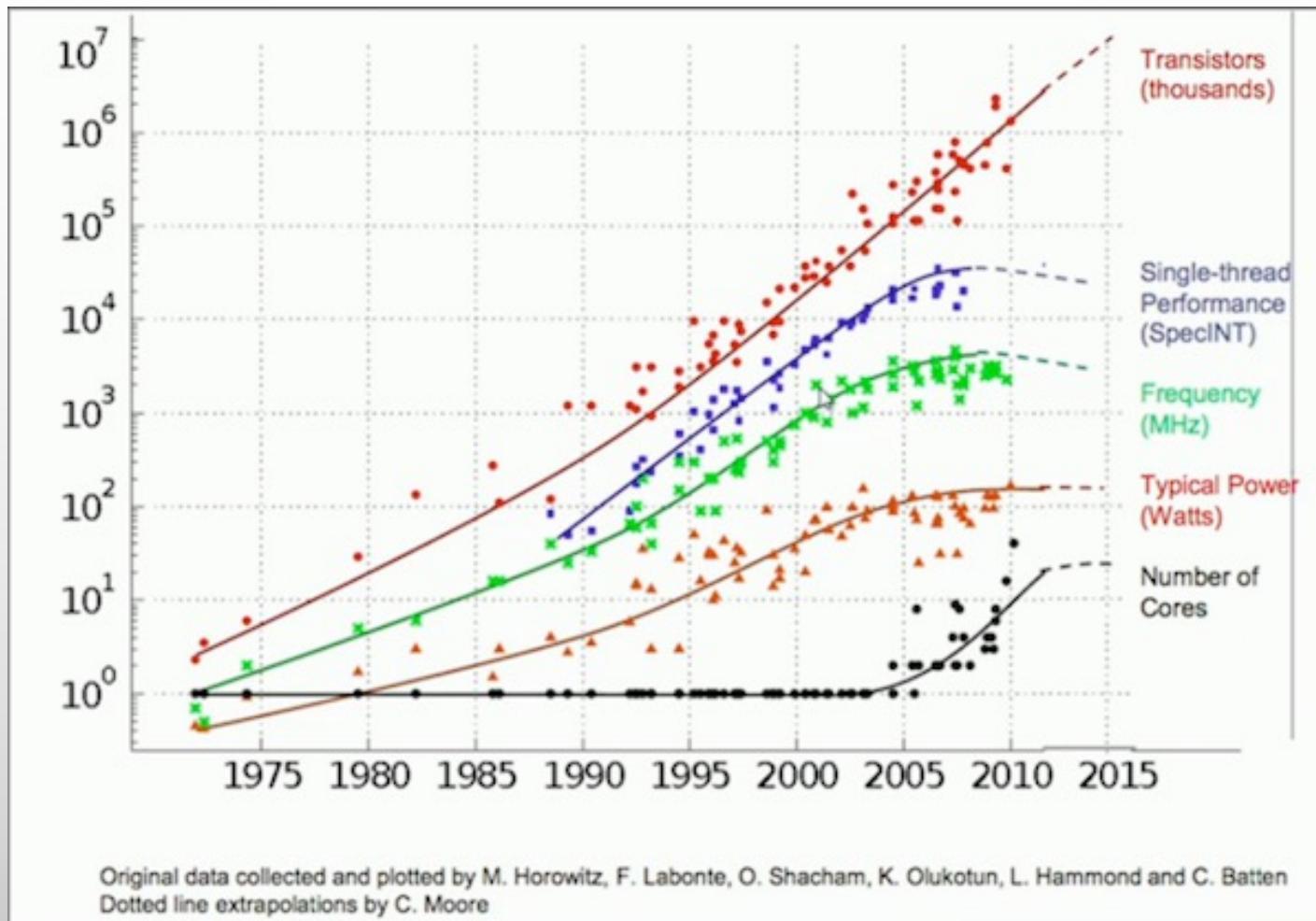


Etude : radar de recul



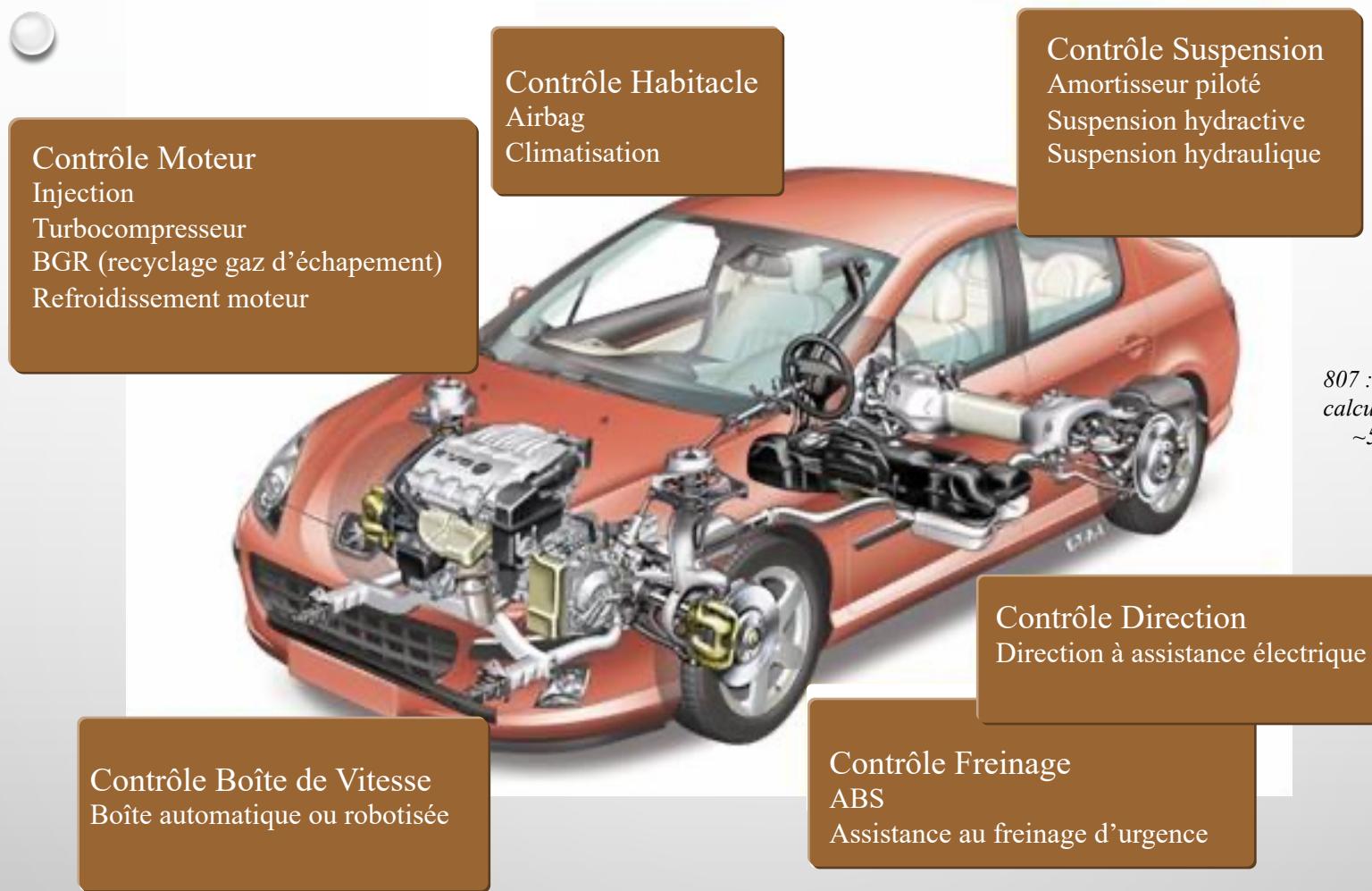
SYSTÈMES NUMÉRIQUES

Systèmes numériques



→ Système monopuce = System On a Chip (SoC)

Applications transports



→ Une Mercedes classe S : 80 processeurs

→ 27-30% du coût d'un véhicule, s'oriente vers 40% du coûts (2020) [source PSA]

Solutions architecturales

Machine programmable

ASIP
(Application Specific Instruction set Processor)

Circuit configurable

ASIC
(Application Specific Integrated Circuit)

DSP

mC

DSP

mC

FPGA

EPLD

PLA

Custom

Semi-Custom

Processeurs généraux

Processeurs multimédias

VLIW

RISC

Circuits sur mesure

Circuits précaractérisés

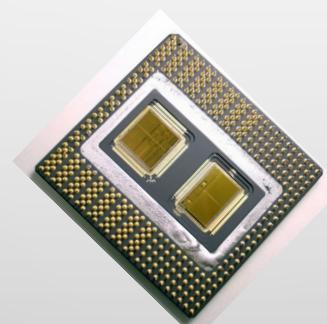
Circuits prédiffusés

Full Custom

Standart cell

Gate array

Sea of gates



VLIW : Very Long Instruction Word

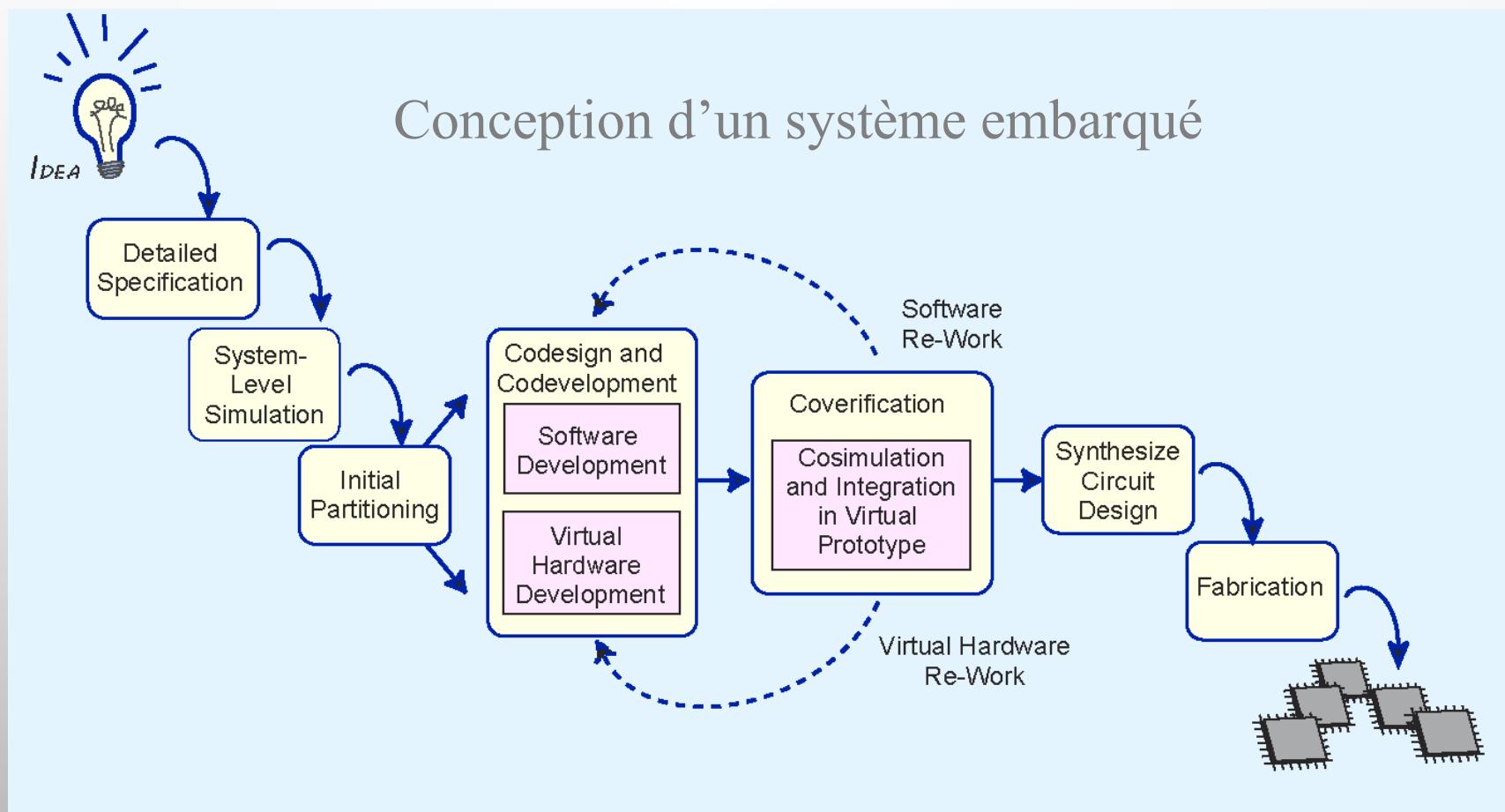
EPLD : Erase Programmable Logic Device

PLA : Programmable Logic Array

Circuits programmables

Il existe deux grandes familles de circuits logiques programmables:

- ✓ Les mémoires programmables
- ✓ Les PLD (programmable logic device)



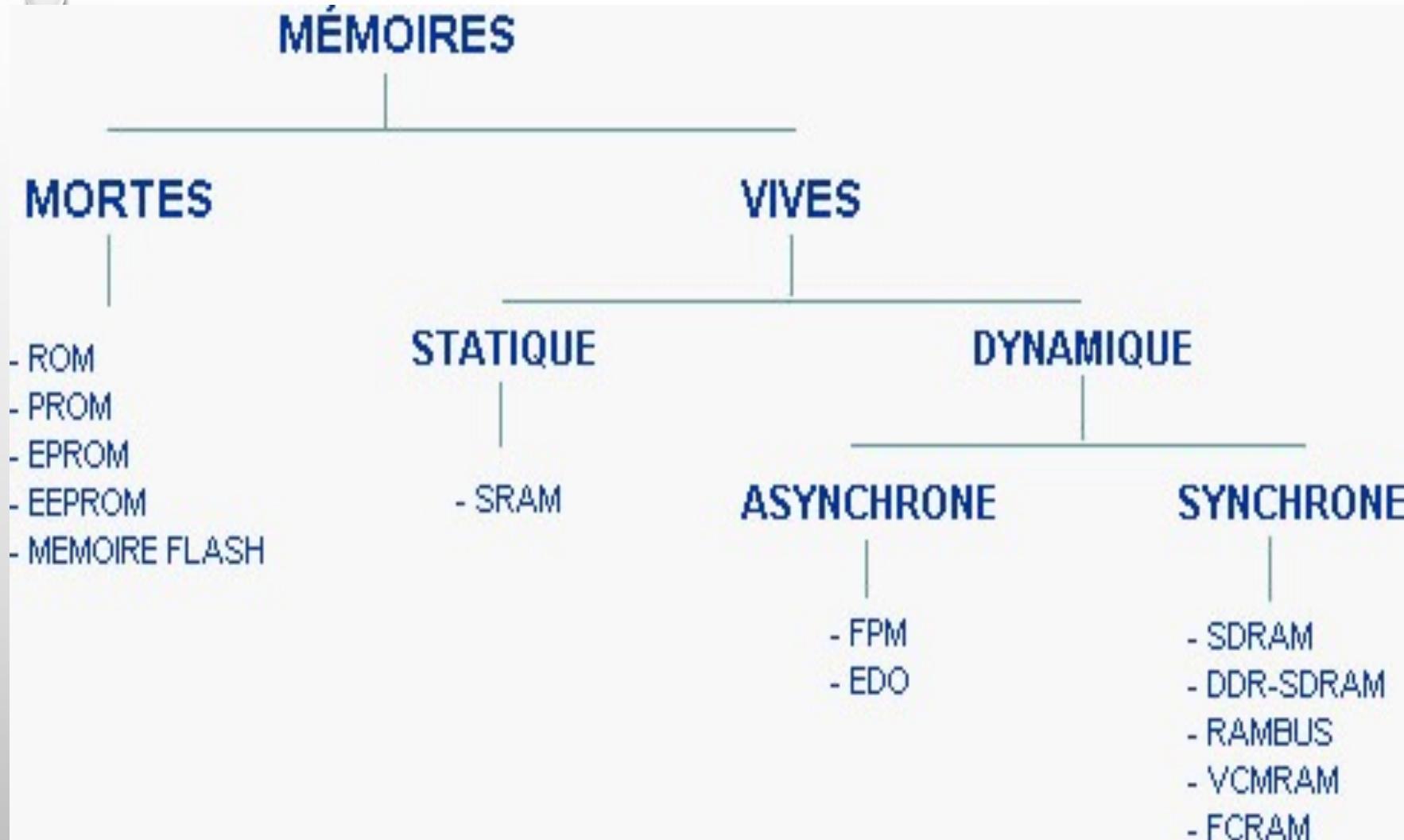
Rappels : la mémoire

Une mémoire est un élément de stockage d'information

Les bits stockés sont organisés sous forme de matrice:

- La dimension de la mémoire est donnée par le nombre de lignes multiplié par la longueur de la ligne
- Chaque ligne de la mémoire est appelée un mot. Elle est identifiée par une adresse (numéro de la ligne)
- Le nombre de lignes est toujours une puissance de deux
- Deux opérations sont possibles, sur un mot complet :
 - La lecture (**read**)
 - L'écriture (**write**)

Classification de la mémoire



La mémoire vive

Random Acces Memory, mémoires vives:

Des mémoires contenant des programmes et des données →
Information disponible sous tension

2 types de RAM: StaticRAM ET DynamicRAM

Statique Ram: Un bit = une bascule D (4 transistors)

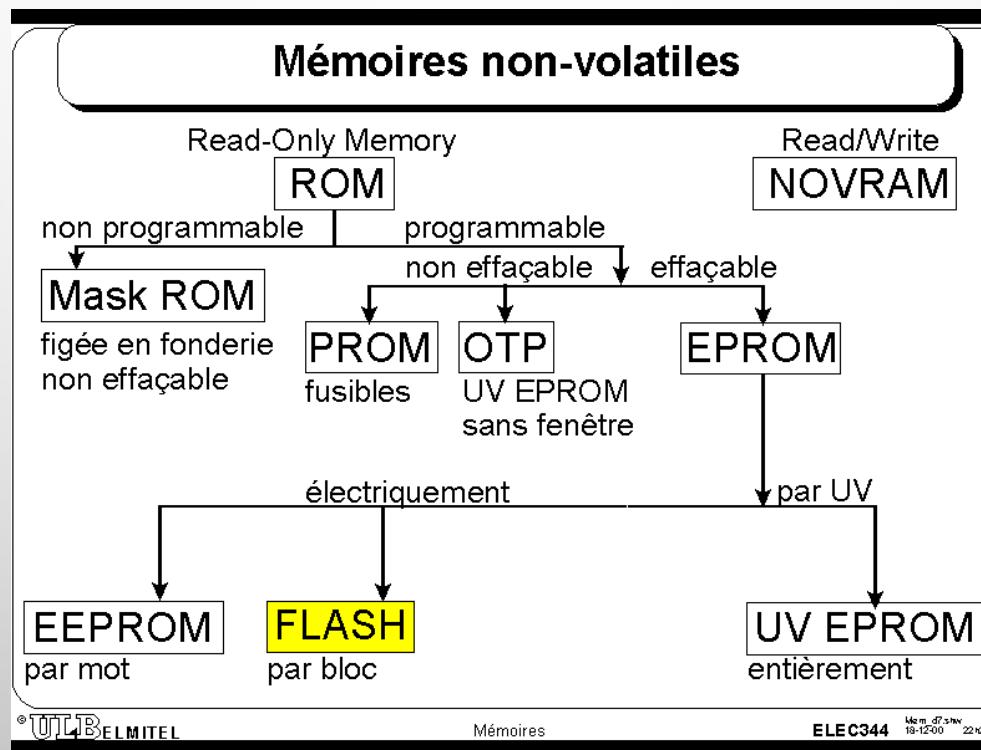
Dynamique Ram : 1bit = une capacité+1 transistor

Les mémoires mortes: read only memory

Les données ne peuvent être que lues

L'écriture se fait soit lors de la fabrication ou nécessite un matériel spécialisé.

La donnée est retenue même en absence du courant



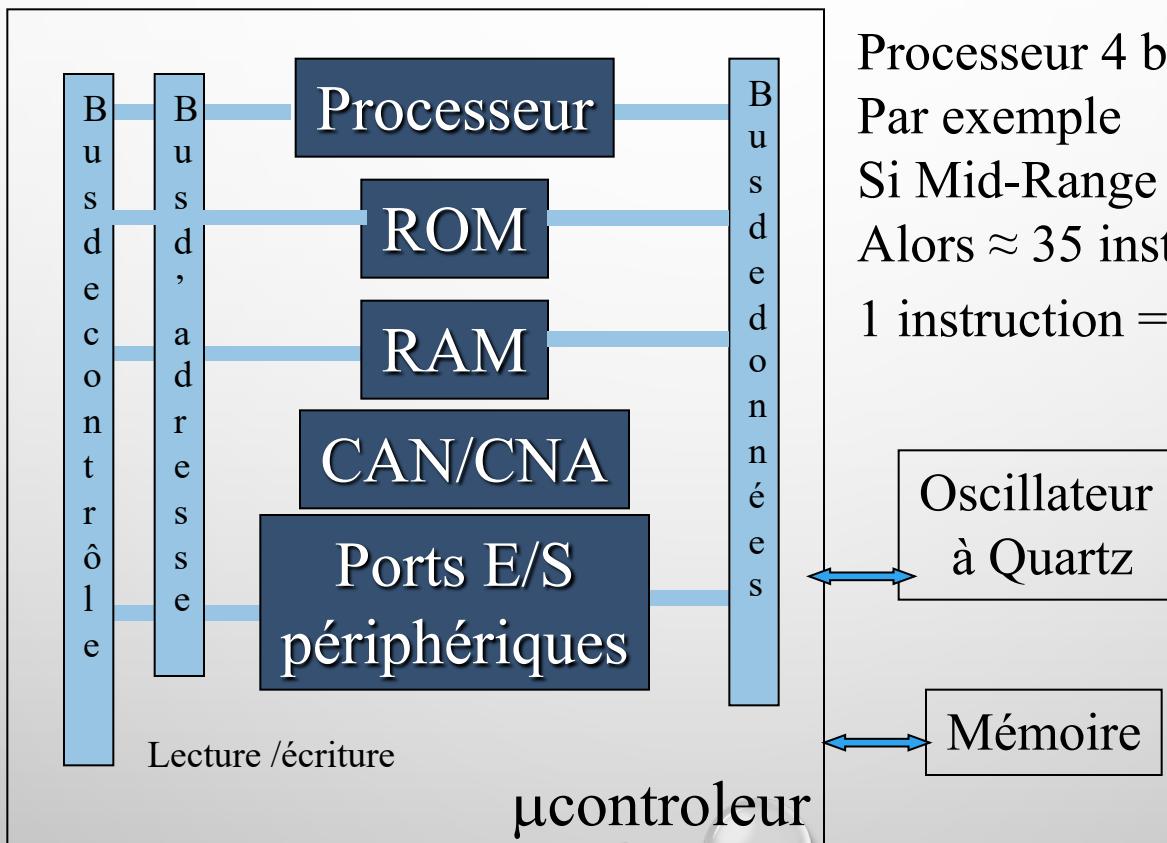
Circuits programmables

- Microcontrôleurs
 - Système Intégré
- Processeurs : Système avec Pérophériques
- DSP Digital Signal Processing
 - Processeurs dédiés au traitement du signal
- Processus RISC Reduced Instruction Set Computer ou CISC Complex Instruction Set Computer

Circuits programmables

Microcontrôleurs: haut degré d'intégration

Unité de traitement de l'information de type microprocesseur
+ périphériques internes



Processeur 4 bits à ...32 bits
Par exemple

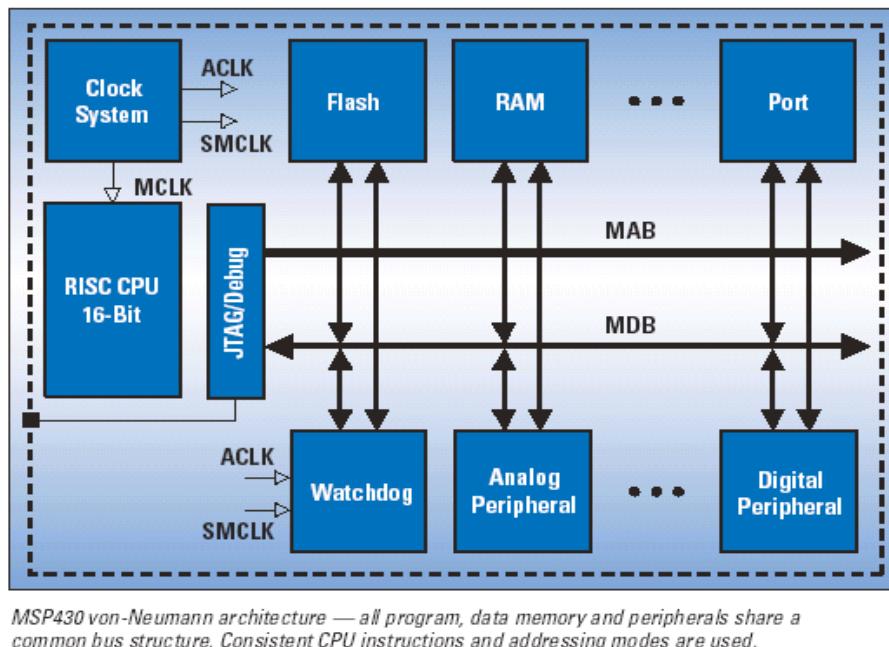
Si Mid-Range & architecture RISC
Alors ≈ 35 instructions

1 instruction = 1 cycle ($\approx 1\text{MIPs}$)

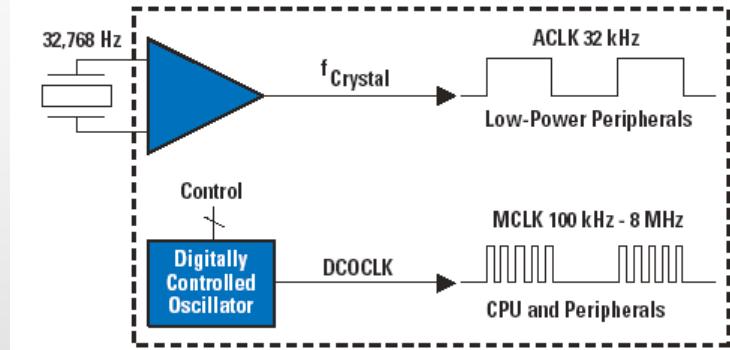
Circuits programmables

Microcontrôleur MSP 430

Modular Architecture

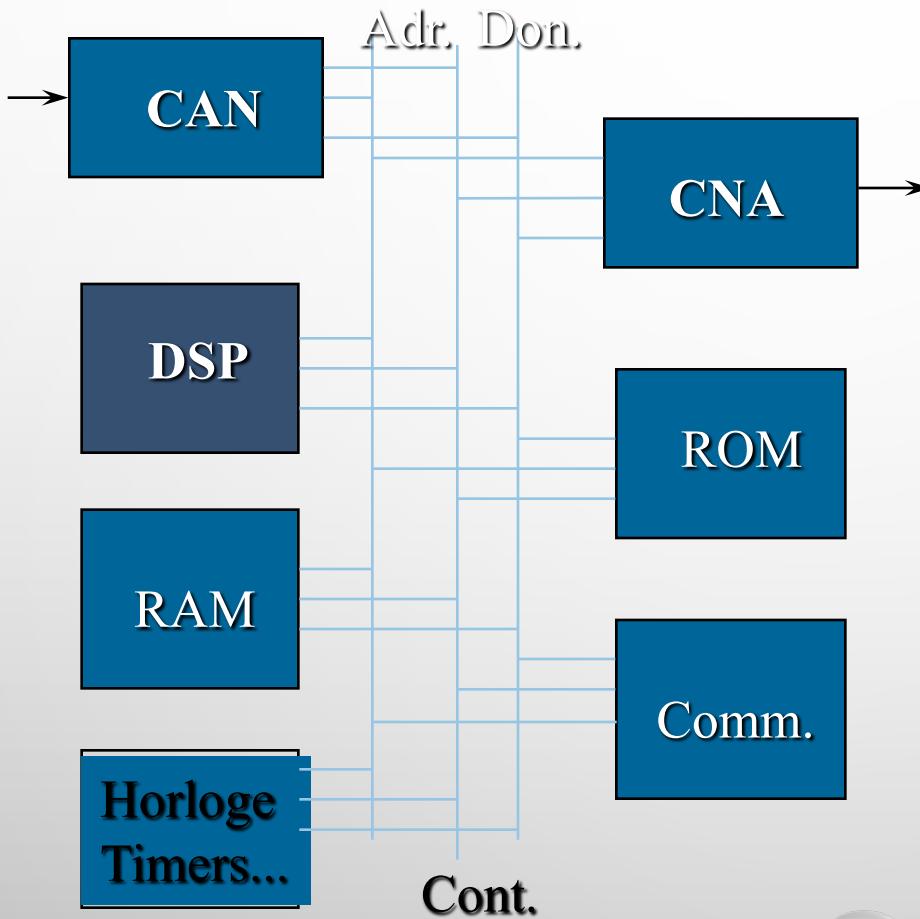


Multiple Oscillator Clock System



Circuits programmables

DSP: optimisé pour le traitement numérique du signal



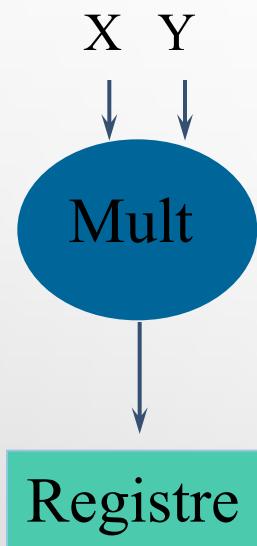
- 2 familles de DSP : virgule fixe ou flottante
- Adressage spécifique
- Pipeline
- parallélisme d'instruction:
- Multiplication
- Instructions plus complexes
- ...

Circuits programmables

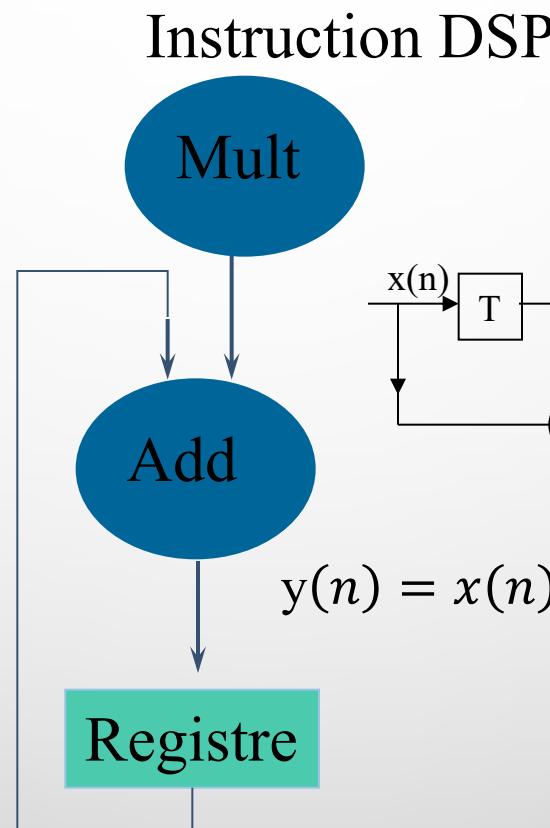
DSP: optimisé pour le traitement numérique du signal

- Instructions plus complexe : exemple

Classique

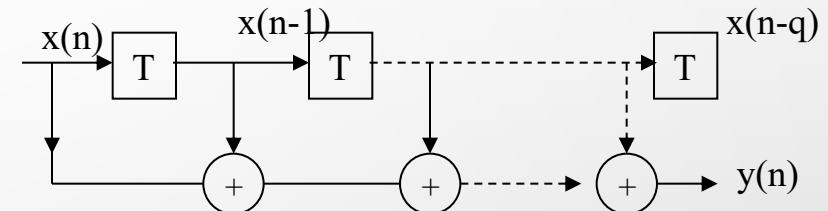


2 instructions



1 instruction MAC

Application:



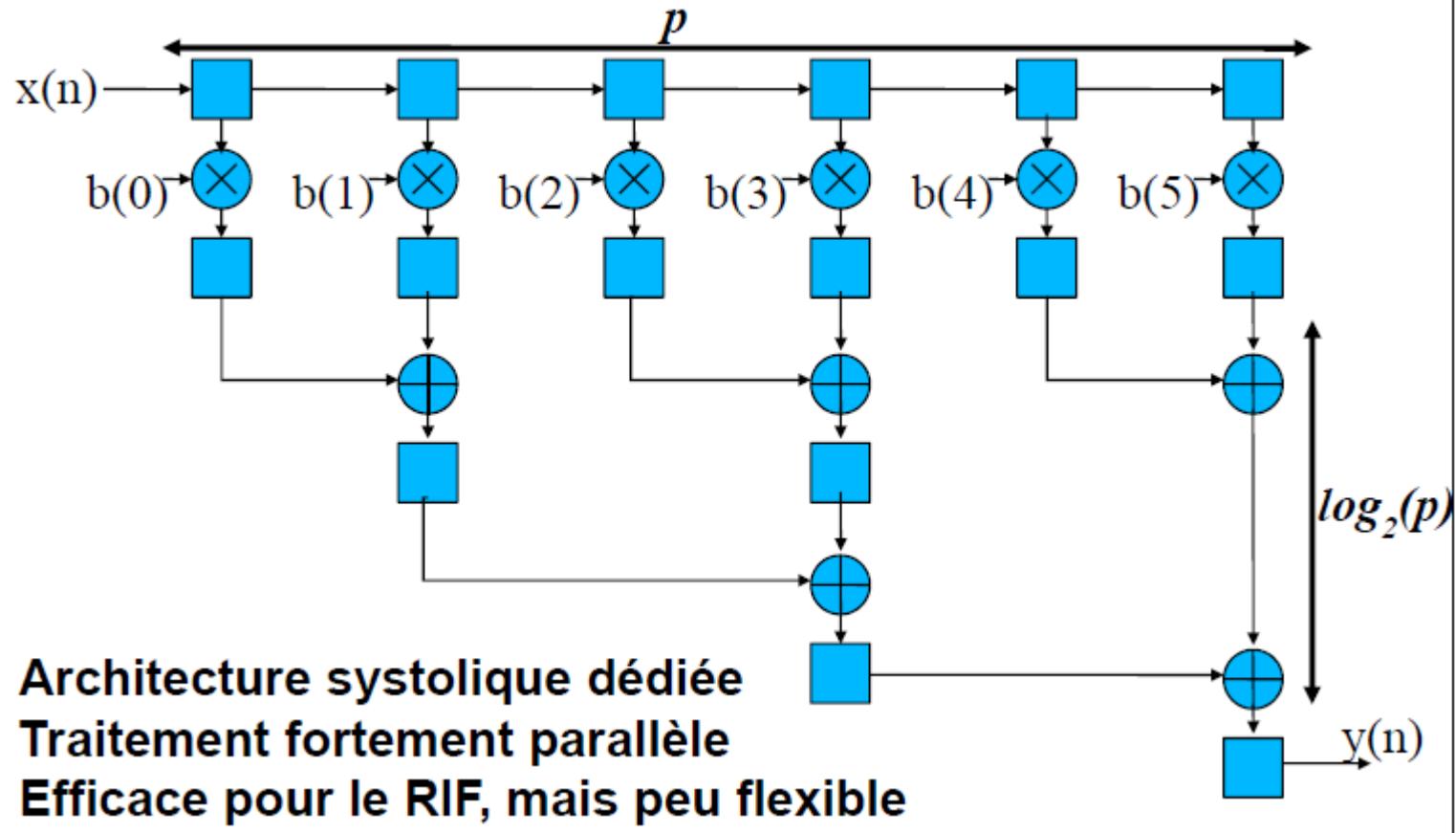
$$y(n) = x(n) + x(n - 1) + \dots + x(n - q)$$

Circuits programmables

DSP: optimisé pour le traitement numérique du signal

- Instructions plus complexe : exemple

$$y(n) = \sum_{i=0}^{P-1} b(i)x(n-i)$$

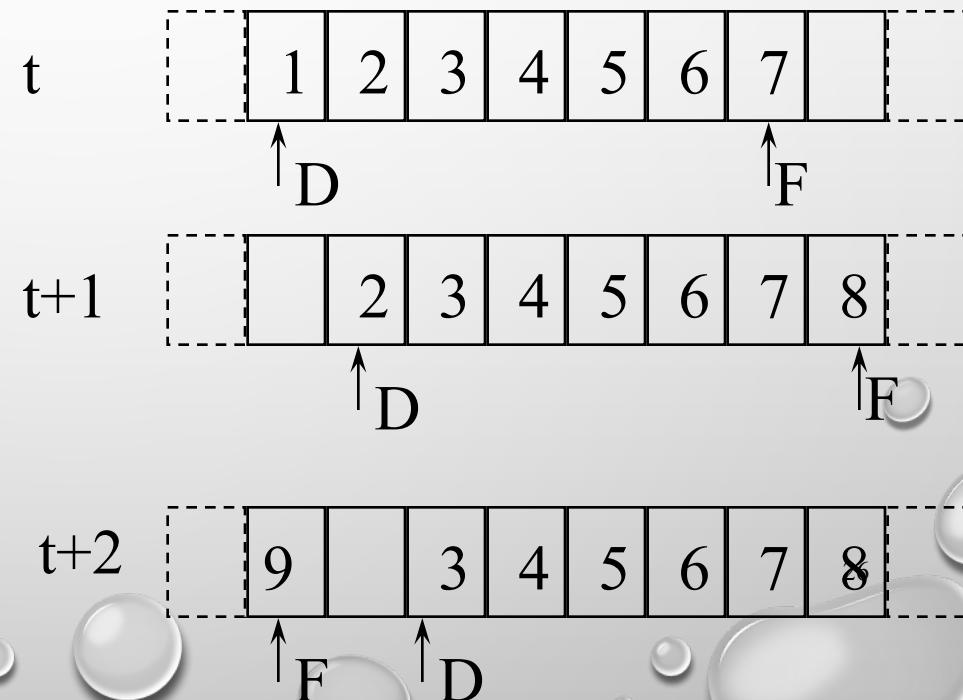


Circuits programmables

- DSP: optimisé pour le traitement numérique du signal
- Instructions plus complexe : Adressage circulaire

But: éviter l'opération d'écriture $x(i)=x(i-1)$ dans le filtre RIF

Principe: utiliser des pointeurs mobiles pour repérer le début et la fin des données



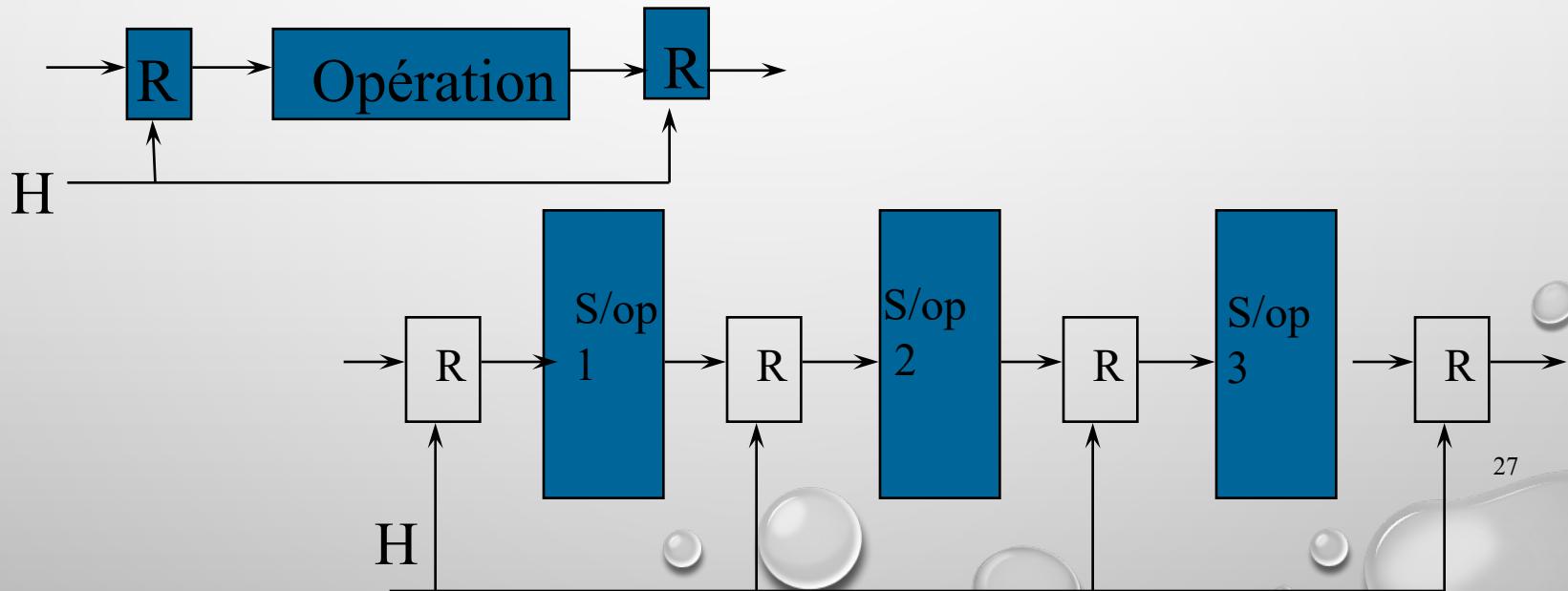
Circuits programmables

- DSP: optimisé pour le traitement numérique du signal

- Instructions plus complexe : Pipeline

Principe du pipeline

Découper une opération en N sous-opérations (S/op) et Exécuter les N sous-opérations en parallèle (sur des données différentes)



Circuits programmables

- DSP: optimisé pour le traitement numérique du signal
 - Instructions plus complexes : Pipeline

Principe du pipeline

Exécution d'une instruction

1. Lecture de l'instruction en mémoire programme (*OpCodeFetch*)
2. Décodage de l'instruction
3. Lecture ou écriture d'une opérande en mémoire de donnée
4. Exécution d'une opération arithmétique ou logique

Cycle	1	2	3	4	5	6	7	8
I/F	I1				I2			
Dec		I1				I2		
R/W			I1				I2	
Exec				I1				I2

Classique

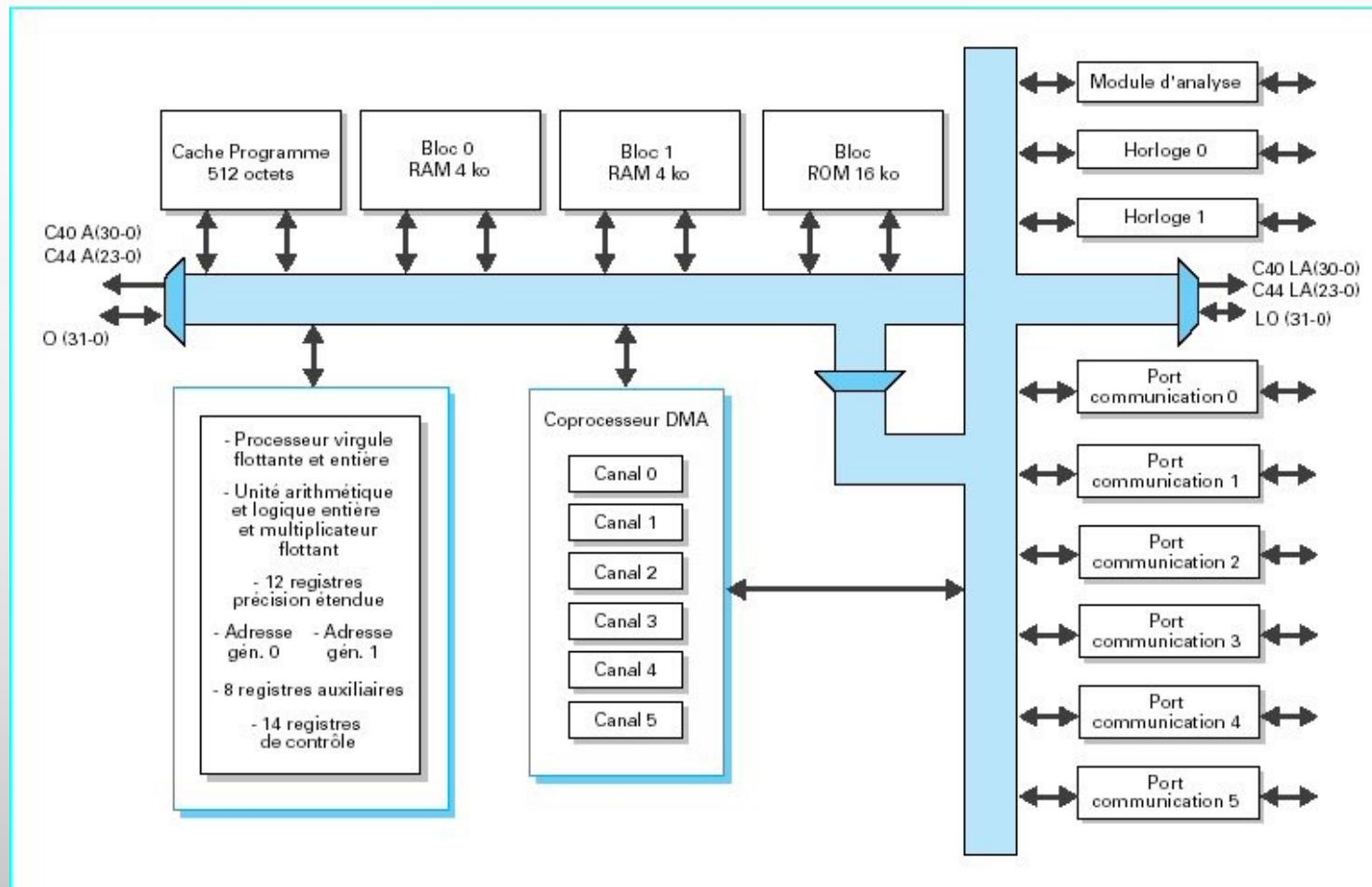
Cycle	1	2	3	4	5	6	7	8
I/F	I1	I2	I3	I4	I5	I6	I7	I8
Dec		I1	I2	I3	I4	I5	I6	I7
R/W			I1	I2	I3	I4	I5	I6
Exec				I1	I2	I3	I4	I5

Pipeline

- La vitesse d'exécution d'un programme est multipliée par N (ici 4) nombre d'étage du pipeline
- En apparence, une instruction est exécutée à chaque cycle horloge
- La durée réelle d'exécution d'une instruction est de N cycles horloge

Circuits programmables

DSP: optimisé pour le traitement numérique du signal



Circuits configurables

Circuits logiques configurables et/ou reconfigurables

- PLD, CPLD Programmable Logic Device or Complex

PLD = Structure de portes logiques

CPLD= Structure de PLD

- FPGA : Field Programmable Gate Array

Reconfigurable → prototypage

coprocesseur

- ASIC : Application Specific Integrated Circuit

Circuit intégré spécialisé fonctionnalité unique /sur mesure

- Système monopuce= System on a chip (SoC)

Système électronique complet intégré dans une puce

Circuits configurables

Circuits logiques reconfigurables

- PLD, CPLD Programmable Logic Device or Complex

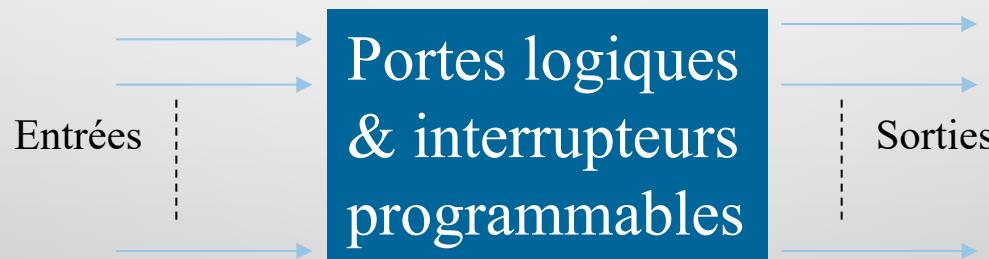
PLD = Structure de portes logiques

CPLD= Structure de PLD

→ circuit monolithique formé de cellules logiques (comportant des fusibles, des antifusibles ou de la mémoire) qui peut être programmé et parfois reprogrammé par l'utilisateur

→ PLD réalise une fonction spécifique: décodeur, driver, compteur...

→ CPLD : circuit composé d'un ensemble de blocs logiques tous reliés à un bloc d'interconnexion (Programmable Interconnect Array)



→ Prototypage

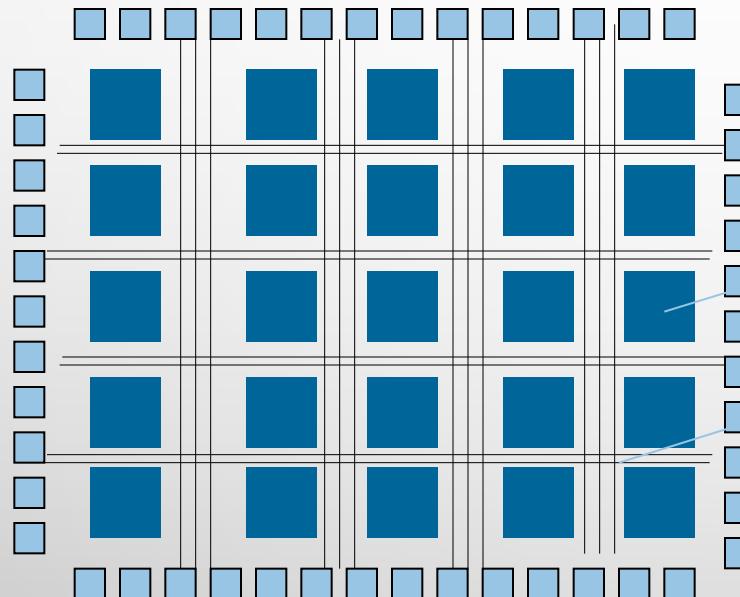
Circuits configurables

Circuits logiques reconfigurables : cellules logiques élémentaires

Connections programmables

Circuit offrant des blocs logiques ainsi que des interconnexions totalement flexibles et qui nécessite un outil de placement routage

Structure générale d'un FPGA



Blocs Entrées/Sorties programmables :
IN, OUT , Registre, HIZ, Triggers de Schmitt

Blocs fonctions

Interconnexions programmables

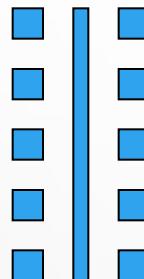
→ La structure est symétrique

→ Le temps d'établissement dépend du routage

Circuits configurables

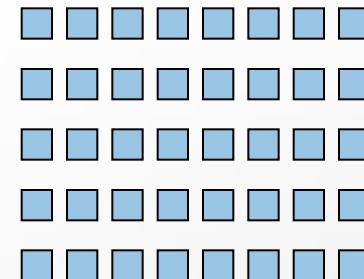
CPLD

Complex Programmable Logic Device



FPGA

Field Programmable Gate Array



Architecture

Combinatoire

Gate array-like
Plus de registres + RAM

Densité Faible à moyenne
 0,5 à 10 K portes logiques

Moyenne à forte
1K à 1 G ‘Gates’

Performances Timing précis
 Environ 250 MHz

Dépend de l’application
>1 GHz

Interconnexion

Matricielle

Incrémentale

Circuits configurables : FPGA

Codage avec un langage HDL

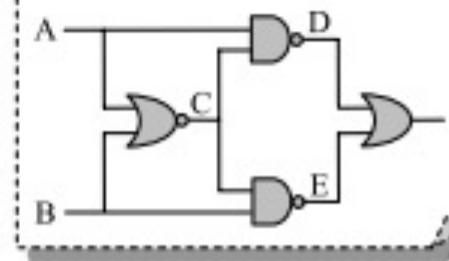
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD.TEXT.all;

entity alg is
  Port ( A,B : in std_logic;
         F : out std_logic );
end alg;

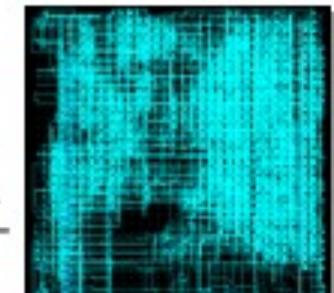
architecture arch of alg is
signal C,D,E : std_logic;
begin
C<=A nor B;
D<=A nand C;
E<=C nand B;
F<=D or E;
end arch;
```

Synthèse

Netlist



Placement & routage



Matrice FPGA



FPGA

Configuration

Bitstream

```
0011000100110000011000  
0110010010010000100010  
0111000001101000001100  
1000111000001110101010
```

Génération du bitstream

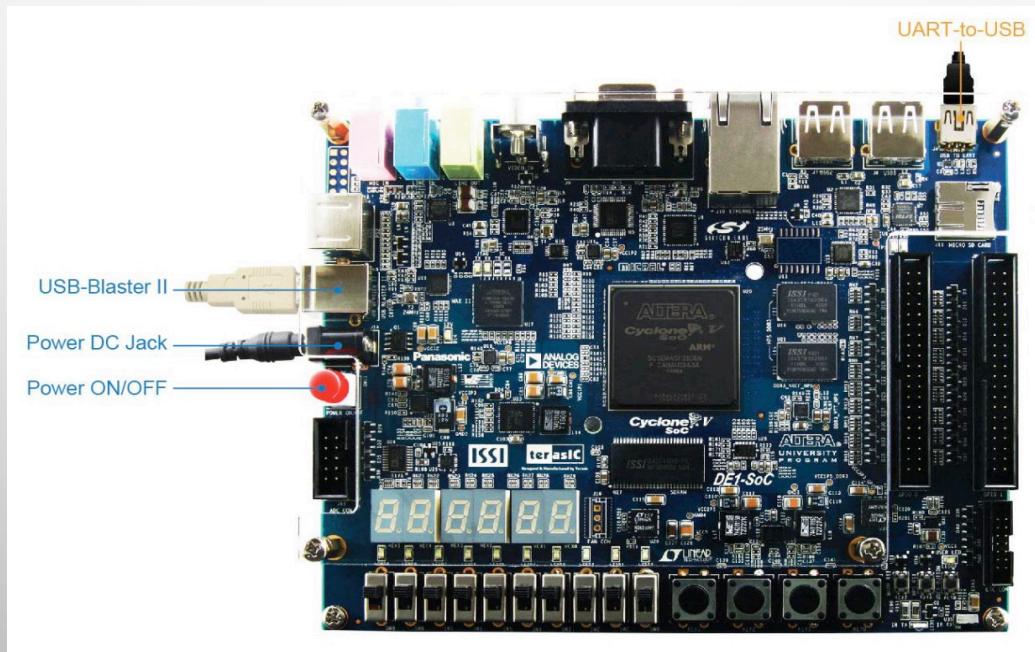
Circuits configurables : FPGA

http://www.techway.fr/fiche-detaillée?id_produit=155

<http://www.agence-nationale-recherche.fr/?Projet=ANR-11-INSE-0002>

Logiciel
Quartus
Description
VHDL

TP : 16heures 4 x 4heures
20heures 5 x 4heures



DSP Builder

Logiciel
Matlab

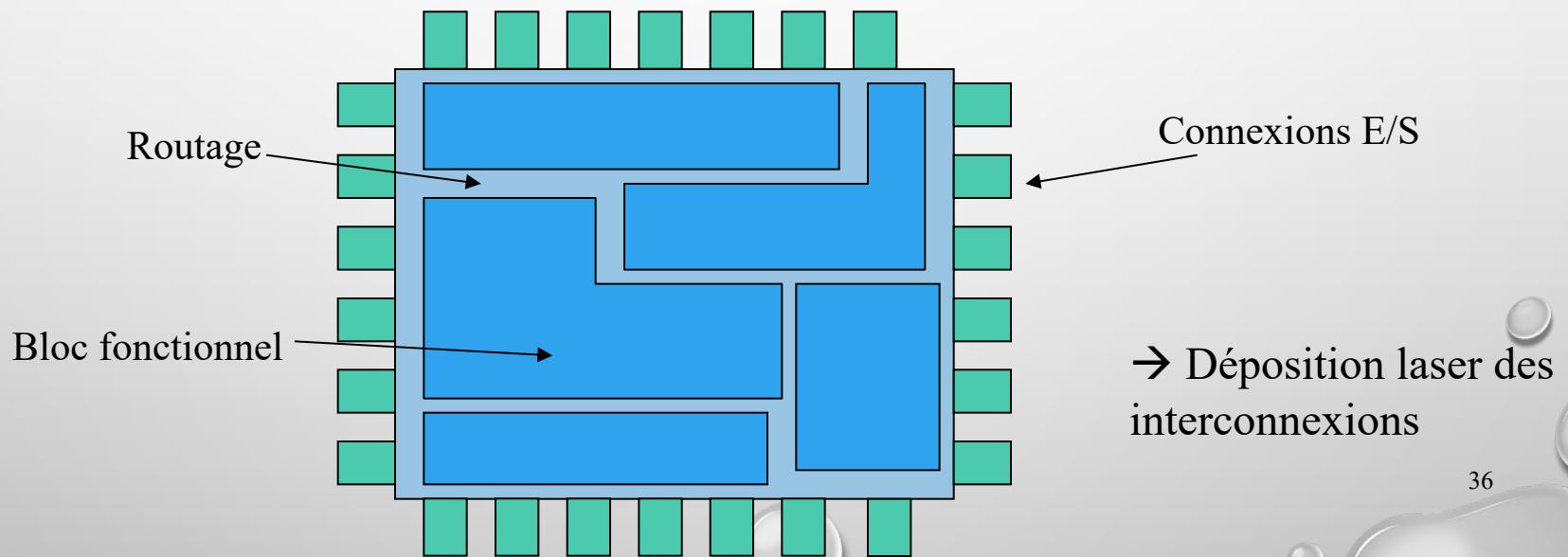
Circuits configurables : ASIC

ASIC (Application Specific Integrated Circuits):

Circuit produit pour un seul client selon les spécifications de ce dernier;

ASSP (Application Specific Standard Product):

Circuit fabriqué selon les technologies ASIC mais vendu tel qu'un composant standard à de nombreux clients



Circuits Configurables : ASIC

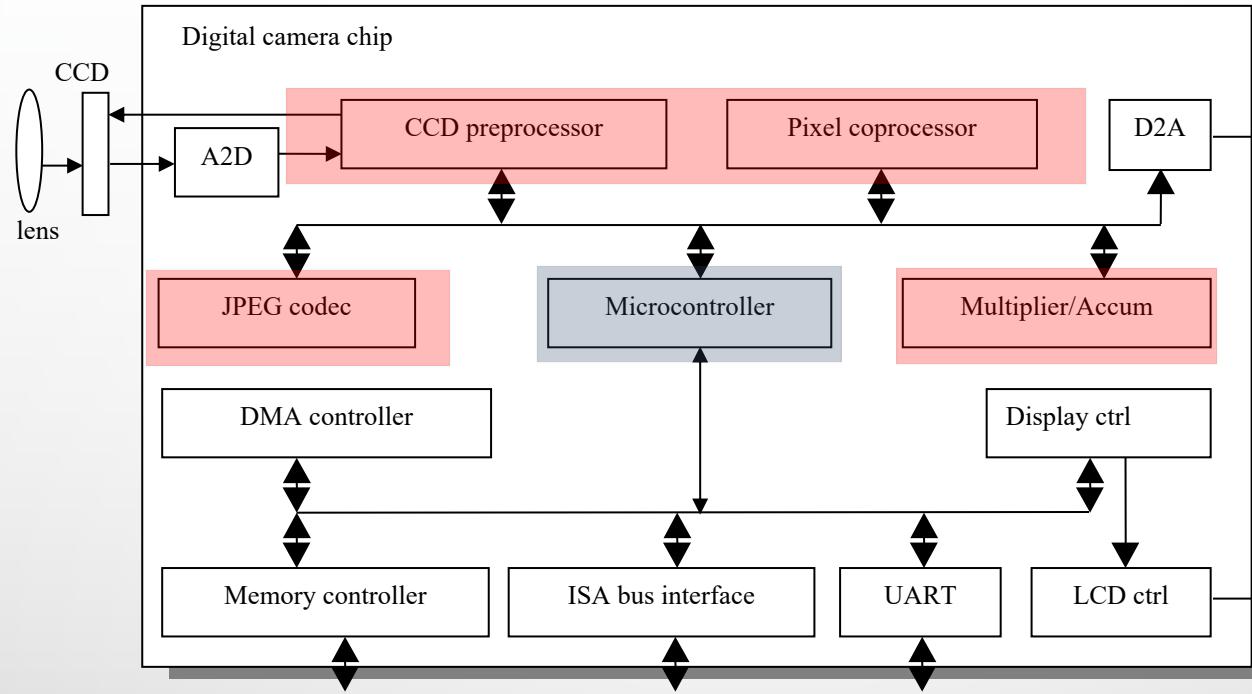
Avantages ASIC :

- réduction du coût final du système
- augmentation de la sécurité du produit
- introduction de fonctionnalités spécifiques
- diminution du nombre de composants du système et réduction de la taille du système
- utilisation efficace de la surface de circuit imprimé
- réduction de la consommation
- augmentation des performances (vitesse)

Désavantages ASIC:

- prix unitaire élevé, développement et conception payée par l'utilisateur
- source potentielle de défauts de conception
- augmentation du temps de développement
- modifications ultérieures difficiles

Systèmes numériques

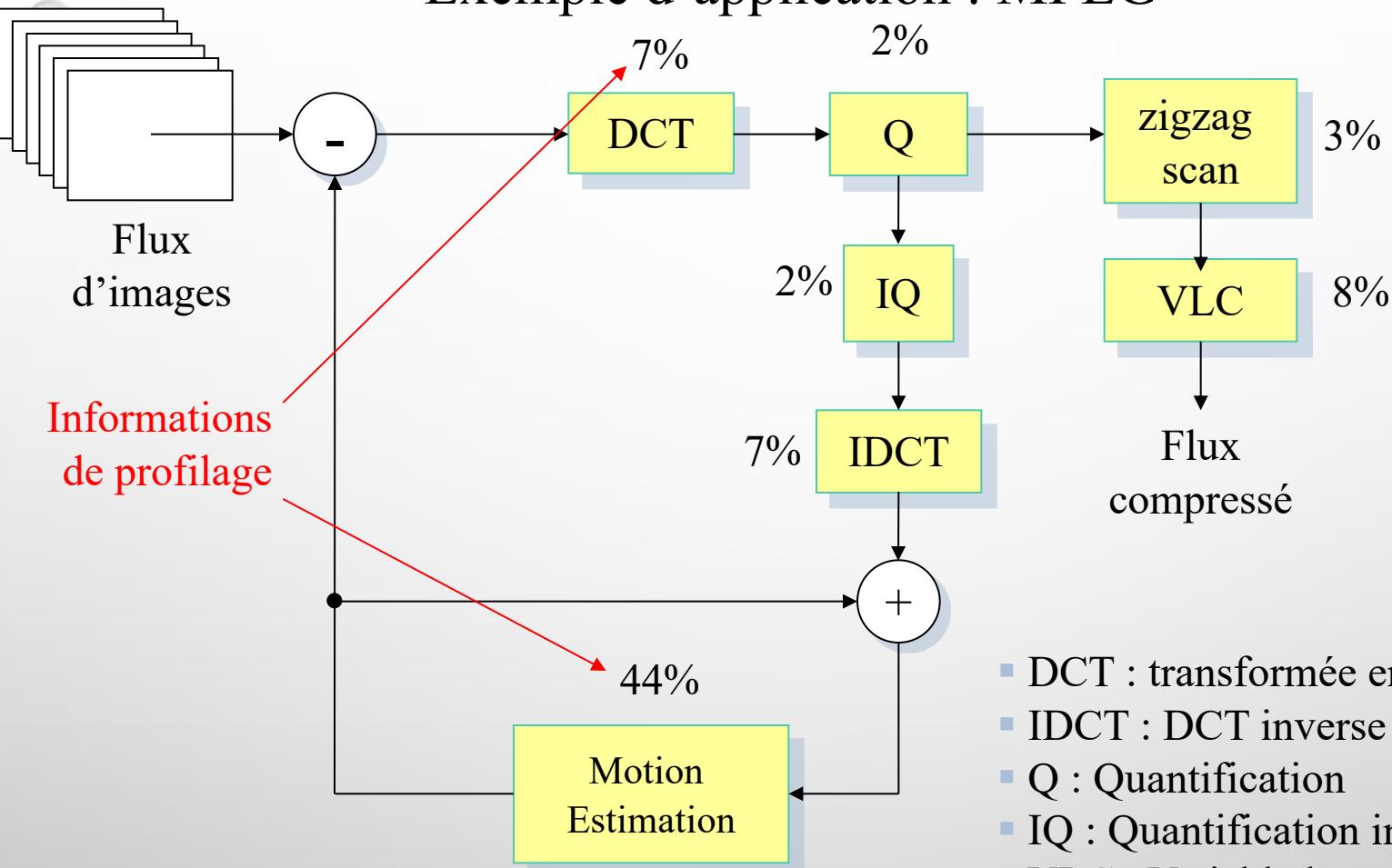


Exemple de structure avec systèmes programmés /systèmes configurés d'Appareil Photo Numérique

Direct Memory Access (DMA) is one of several methods for coordinating the timing of data transfers between an input/output (I/O) device and the core processing unit or memory in a computer.

Systèmes numériques

Exemple d'application : MPEG

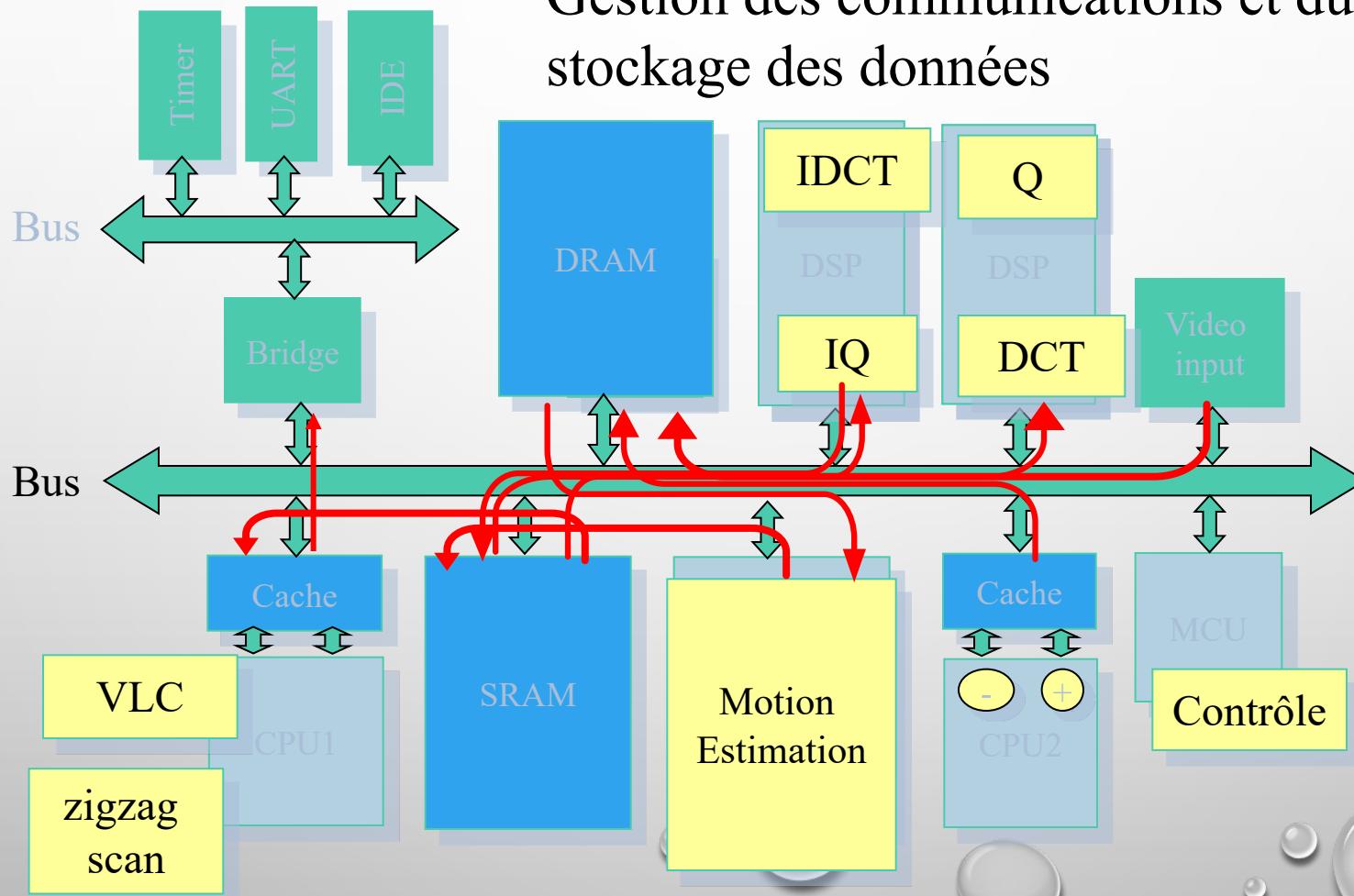


- DCT : transformée en cosinus
- IDCT : DCT inverse
- Q : Quantification
- IQ : Quantification inverse
- VLC : Variable length coding

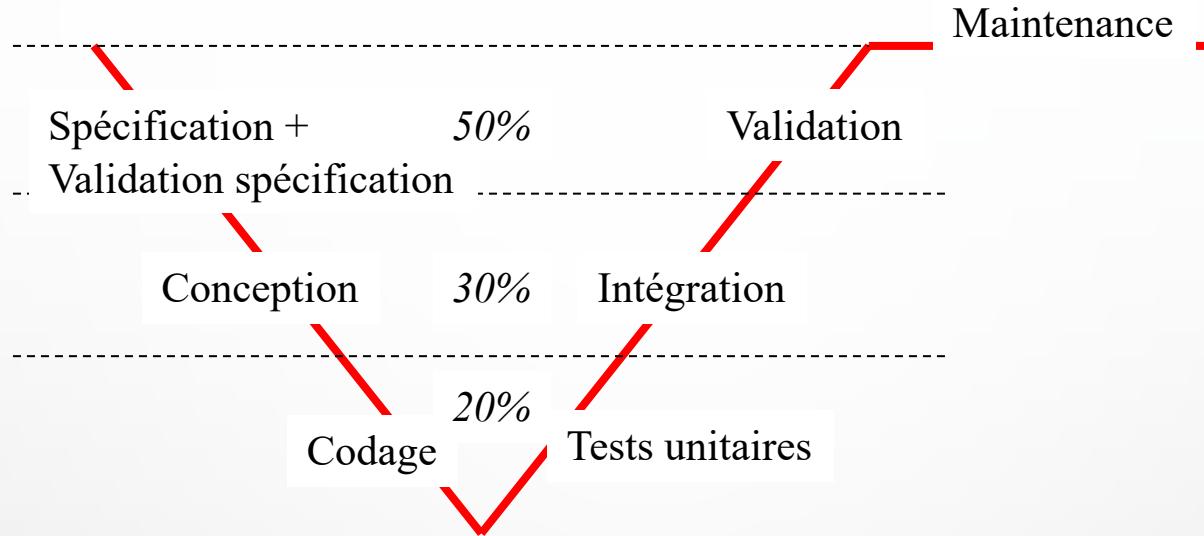
Systèmes numériques

Mise en œuvre

Gestion des communications et du stockage des données



Conception de systèmes

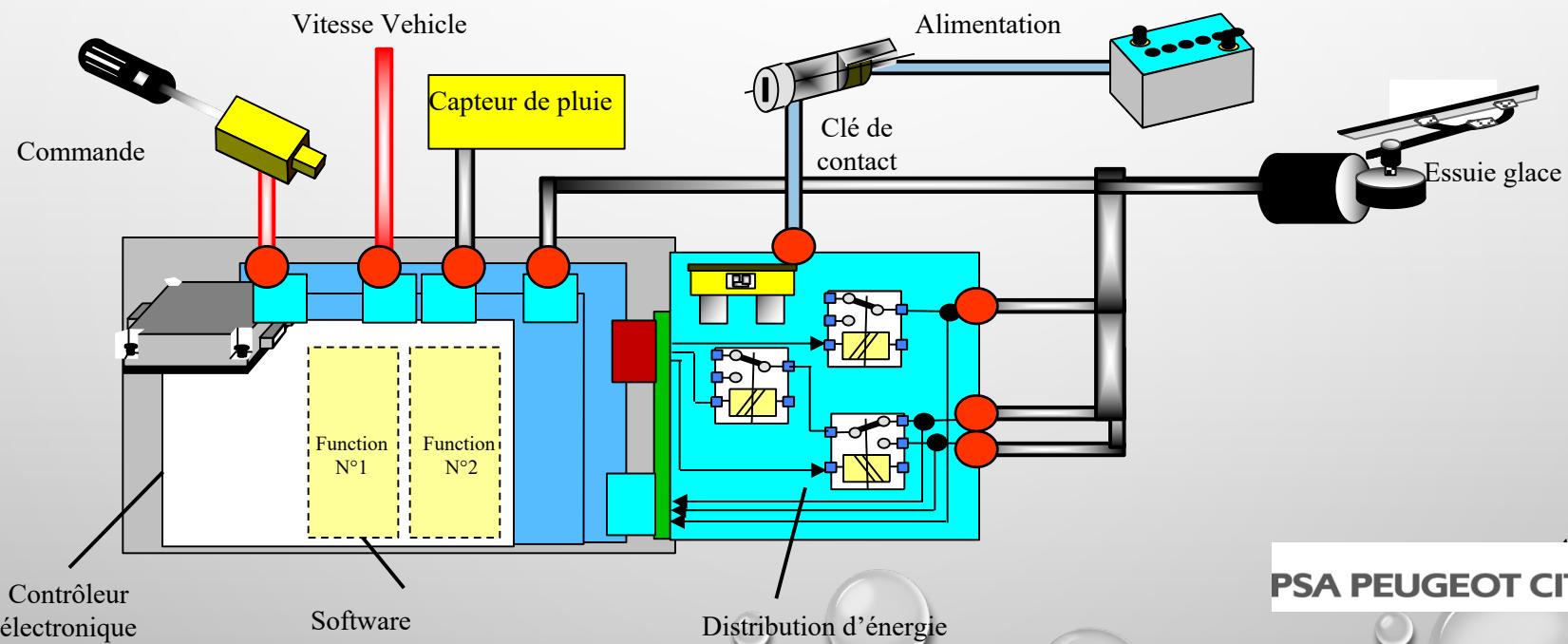


- ▶ Exemple de l'automobile
 - Coûts, volumes, forte imbrication des technologies, ... L'électronique et le logiciel au coût automobile
 - Une électronique véhiculaire « transversale » est fabriquée à 1 million d'unités / an à un coût « pièce » de 40 à 80 Euros et des coûts d'études de plusieurs millions d'Euros
 - **Le coût logiciel reste faible par rapport au coût pièce**; par contre les risques sur les délais sont de plus en plus pesant...
- ▶ Le constructeur doit maîtriser le processus d'intégration
 - c'est une contrainte de plus en plus forte

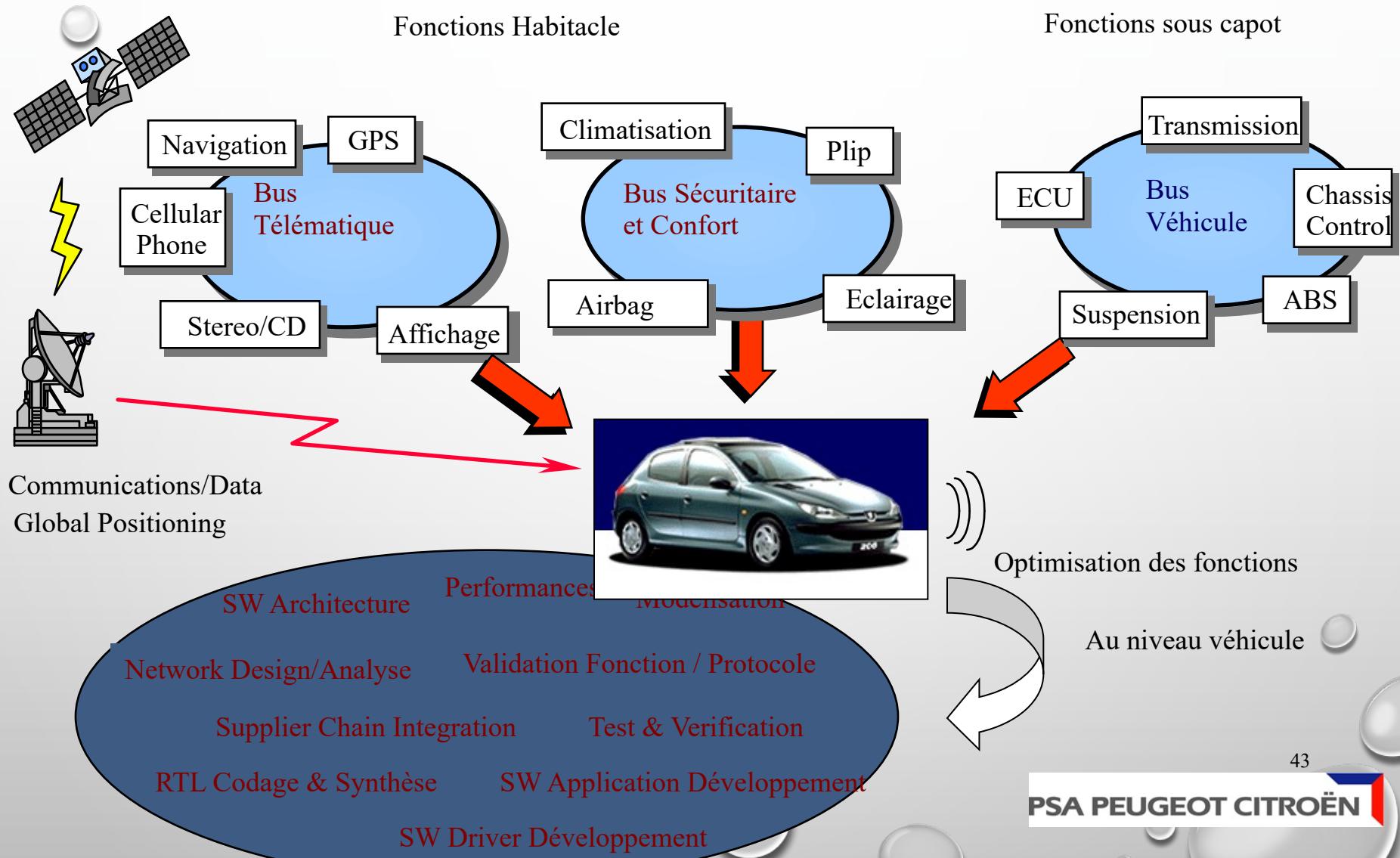
Exemple de fonctionnalité

INTEGRATION D'UNE FONCTION VEHICULE : balayage automatique

- Alimentation et distribution d'énergie
- Commande / capteurs / actionneurs
- Contrôleur Electronique : microprocesseur / hardware interfaces / basic software / application software



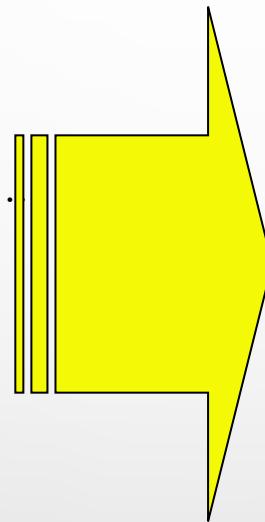
Exemple d'intégration



Véhicules : les contraintes

Les nouvelles fonctionnalités :

- Commandes électriques
 - Accélérateur
 - Freinage
 - Boîte de vitesses
 - Direction....
- Confort
 - Accès mains libre
 - Climatisation automatique...
- Sécurité
 - Airbag intelligent
 - Pré crash
 - Appel d'urgence....
- Télématique
 - Téléphonie
 - Assistance et télédiagnostic
 - Internet....
- Multimédia
 - Audio
 - Vidéo
 - Navigation....



Les conséquences:

- ↑ FLUX D'INFORMATION
- ↑ SÛRETÉ DE FONCTIONNEMENT
- ↑ TEMPS DE REPONSE
- ↑ COMPATIBILITE INFRASTRUCTURE
- ↑ PUISSANCE DE CALCUL

Véhicules : Les évolutions futures

- Augmentation du nombre de calculateurs
 - Besoin d'optimiser l'architecture et de standardiser les fonctions et les interfaces afin de maîtriser les coûts
- Accroissement du couplage entre les fonctions (ESP, CGC, ABS, ...)
 - Besoin de structurer les lois de commandes
- Accroissement de la complexité (x by wire, ...)
 - Besoin de maîtriser la sûreté de fonctionnement afin d'assurer la sécurité des véhicules et de ses occupants.
- Ceci entraînant une diversité des différentes variantes à maîtriser

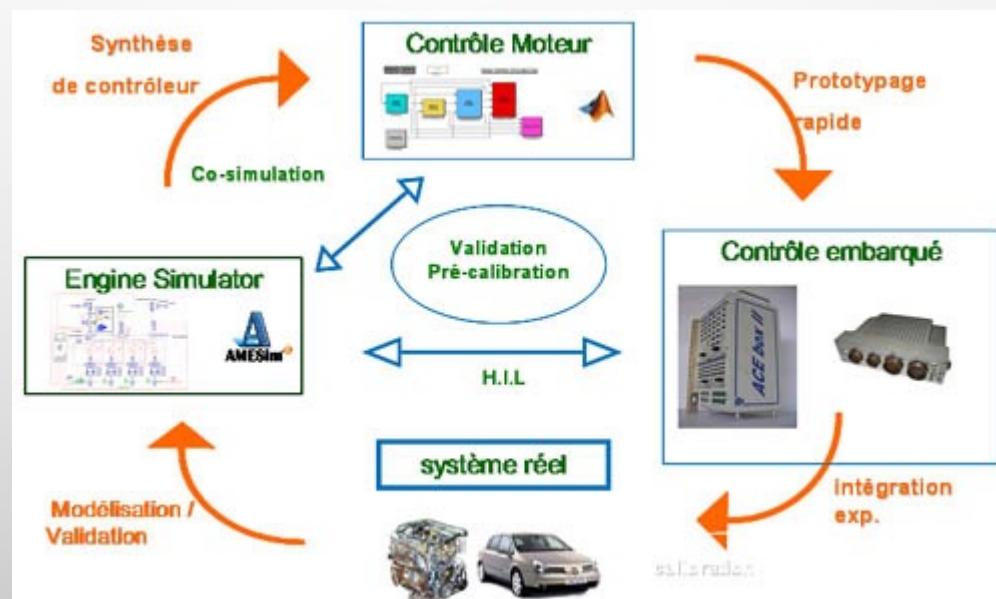
Véhicules : Les évolutions futures

- ▶ L'évolution des besoins et les contraintes réglementaires accélèrent l'introduction de l'électronique dans l'automobile.
- ▶ L'électronique devient visible et incontournable.
- ▶ Avec l'électronique, le logiciel est présent dans l'automobile.
- ▶ Une évolution des architectures et des réseaux est en cours mais des mutations plus profondes sont à venir.
- ▶ L'optimisation doit être conduite au niveau fonctionnel véhicule et être déclinée sur les composants, les architectures électriques, électroniques et la gestion de l'énergie.
- ▶ Les contraintes de compatibilité et sûreté de fonctionnement orienteront les choix futurs.

Conception et prototypage

La simulation Hardware-In-the-Loop (HIL)

- ▶ C'est une méthode de simulation caractérisée par l'association de véritables composants, connectés à une partie temps-réel simulée



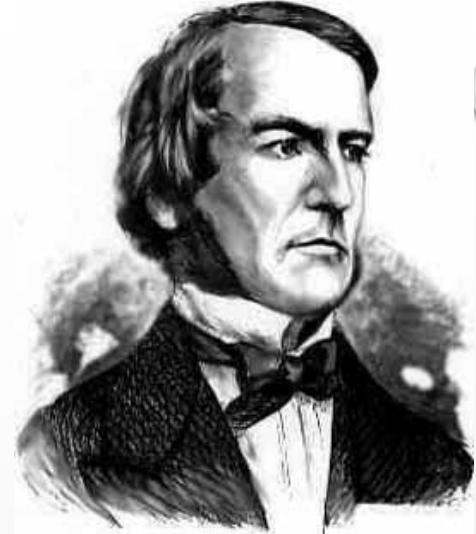
Conception et prototypage

- Les avantages offerts par la simulation hardware-in-the-loop sont multiples
 - La conception et les tests du système de contrôle peuvent être faits sans le système réel
 - Les conditions d'essais sur système de contrôle matériel peuvent être poussées à l'extrême (exemple : haute/basse température, fortes accélérations et choques mécaniques, compatibilité électromagnétique).
 - Il est possible d'effectuer des essais sur les effets de défauts et pannes des actionneurs, capteurs et ordinateurs sur l'ensemble du système
 - Opérations et essais en conditions extrêmes et dangereuses
 - Expériences reproductibles et répétables à souhait.
 - Opérations facilitées avec différentes interfaces hommes-machines (conception de cockpit et formation des opérateurs).
 - Économie de temps et d'argent dans le processus de développement.₄₈

ALGÈBRE DE BOOLE

Origines

- George Boole (1815 – 1864)
- Mathématicien anglais
- 2 ouvrages fondateurs:
 - "The mathematical analysis of logic", 1847
 - An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities", Boole 1854
- Ces travaux sont inexploités pendant environ 70-80 ans
- Applications technologiques: Shannon (~1930)
- L'algèbre de Boole est l'outil mathématique qui permet d'établir une relation entre les sorties et les entrées d'un système logique (synthèse du système).



Algèbre binaire

- Bases de l'algèbre de Boole → 2 états => systèmes logiques
- États logiques : (purement symbolique)
 - 0 et 1,
 - Vrai et Faux,
 - H et L
 - Raymonde et Robert !!!
- variable logique : symbole pouvant prendre comme valeur des états logiques (a,b,c, ou ...)
- fonction logique F: expression de variables et d'opérateurs
 $x=f(a,b,c\dots)$

Éléments de base

- **Variables d'entrée**

Les variables d'entrée sont celles sur lesquelles on peut agir. Ce sont des variables logiques indépendantes.

- **Variables de sortie**

Variables contenant l'état de la fonction après application des opérateurs logiques sur les variables d'entrée.

- **Simplification d'une fonction logique**

Trouver la représentation (l'écriture) la plus simple de la fonction réalisée:
algèbre de Boole

Opérations fondamentales

Algèbre de Boole sur $[0,1]$ = algèbre binaire

Structure d'algèbre de Boole

- 2 lois de composition interne
- 1 application unaire

2 Lois de Composition Interne : ET, OU

OU ou OR (Union, Addition Logique) $s = a + b$

ET ou AND (Intersection, multiplication logique) $s = a \cdot b$

Application unaire : $s = \bar{a}$

Not (complémentation, inversion)

→ « a barre » ou « non a »

Autres opérations : Par combinaisons de ces opérations de base :
Ou exclusif, NI (NOR), NAND,....

Inversion

- Une variable booléenne ne peut prendre que deux états (0 et 1)
- L'inversion d'une variable est l'autre code de la variable
- Soit a variable booléenne dite sous forme normale
- A sous sa forme complémenté: l'inversion de a dite a barre : \bar{a}

a	\bar{a}
0	1
1	0

Fonction ET

- Le ET de deux variables booléennes est une autre variable booléenne qui prend l'état 1 si et seulement si les deux variables sont à l'état 1.
- Notation $a.b$ ou plus simplement ab

a	0	1	0	1
b	0	0	1	1
ab	0	0	0	1

$$x = a.b$$

Table(au) de vérité
de la fonction ET

Fonction OU

- Le OU de deux variables booléennes est une autre variable booléenne qui prend l'état 1 si au moins une des variables est à l'état 1.
- Notation $a+b$ (ou et non « plus »)

a	0	1	0	1
b	0	0	1	1
$a+b$	0	1	1	1

$$x = a + b$$

Table(au) de vérité de la fonction OU

Propriétés

- **Commutativité**
 $\forall(a, b) \in E^2$
 $a+b = b+a$
 $a.b = b.a$
- **Associativité**
 $\forall(a, b, c) \in E^3$
 $a+(b+c) = (a+b)+c$
 $a.(b.c) = (a.b).c$
- **Distributivité**
 $\forall(a, b, c) \in E^3$
 $a.(b+c) = a.b + a.c$
 $a+(b.c) = (a+b).(a+c)$

Idempotence

$$\begin{aligned}\forall(a) \in E \\ a+a = a \\ a.a = a\end{aligned}$$

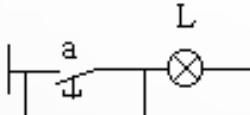
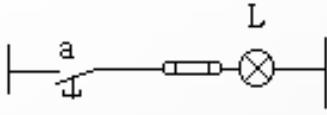
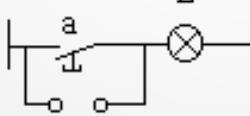
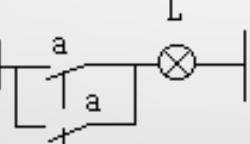
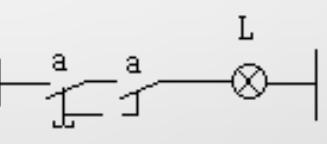
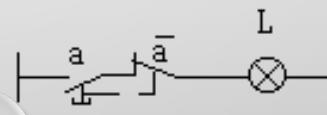
Absorption

$$\begin{aligned}\forall(a, b) \in E^2 \\ a+a.b = a \\ a.(a+b) = a\end{aligned}$$

Involution

$$\overline{\overline{a}} = a$$

Propriétés : applications

Sommes logiques	Produits logiques
 $a + 1 = 1$	 $a \cdot 1 = a$
 $a + 0 = a$	 $a \cdot 0 = 0$
 $a + a = a$	 $a \cdot a = a$
 $a + \bar{a} = 1$	 $a \cdot \bar{a} = 0$

Propriétés

■ Elément neutre

$$a+0 = a$$

$$a \cdot 1 = a$$

■ Elément absorbant

$$a+1 = 1$$

$$a \cdot 0 = 0$$

■ Inverse

$$a + \bar{a} = 1$$

$$a \cdot \bar{a} = 0$$

Théorème de DE Morgan

$$\overline{a + b} = \bar{a} \cdot \bar{b}$$

$$\overline{a \cdot b} = \bar{a} + \bar{b}$$

Théorème de (De) Morgan

■ Pour obtenir l'inverse d'une fonction booléenne:

- ◆ On ramène l'expression à une forme OU ou ET de variables
- ◆ On inverse toutes les variables
- ◆ On remplace la fonction ET par la fonction OU et la fonction OU par la fonction ET

$$\overline{a \cdot b \cdot c \dots q} = \overline{a} + \overline{b} + \overline{c} + \dots + \overline{q}$$
$$\overline{a + b + c + \dots + q} = \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \dots \cdot \overline{q}$$

■ Exemples:

$$\overline{a \cdot b + c} = (\overline{a} + \overline{b}) \cdot \overline{c} = \overline{a} \cdot \overline{c} + \overline{b} \cdot \overline{c}$$

$$a + \overline{bc} + \overline{cd} = \overline{a} \cdot b \cdot c \cdot \overline{\overline{c} \cdot d} = \overline{a} \cdot b \cdot c(c + \overline{d}) = \overline{a} \cdot b \cdot c$$

Propriété d'absorption

• Autres fonctions importantes

Toute fonction logique peut s'écrire avec les opérateurs ET, OU et NON (par définition)

- Groupe d'opérateurs complet (**GOC**) : ensemble d'opérateurs permettant d'exprimer toutes les fonctions logiques
- Trois opérateurs importants et fonction des opérateurs de ce GOC : nand, nor ou exclusif

OU exclusif

- Ou exclusif: fonction de deux variables booléennes qui prend la valeur 1 si uniquement une ou l'autre des variables (mais pas les deux) est égale à 1

a	0	1	0	1
b	0	0	1	1
$a \oplus b$	0	1	1	0

$$x = a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$$

Tableau de vérité

Notation: $x = a \oplus b$ mais aussi $x = a \wedge b$ ou $x = a \text{ xor } b$

OU exclusif

- La fonction « ou exclusif » est:
 - Commutative : $a \oplus [b \oplus c] = [a \oplus b] \oplus c = a \oplus b \oplus c$
 - Distributive : $a \cdot [b \oplus c] = a \cdot b \oplus a \cdot c$
- Le complément de la fonction « ou exclusif » est parfois appelée fonction « coïncidence »

Fonction NOR (NI)

- Fonction qui vaut 1 si et seulement si la totalité des variables est à l'état 0
- Représentation

$$x = \overline{a + b} = \overline{a} \cdot \overline{b}$$

a	0	1	0	1
b	0	0	1	1
x	1	0	0	0

Table(au) de vérité

Fonction NAND

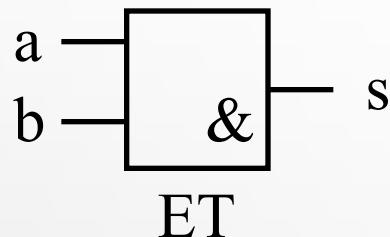
- Fonction qui vaut 1 si une au moins des variables est à l'état 0
- Représentation

a	0	1	0	1
b	0	0	1	1
x	1	1	1	0

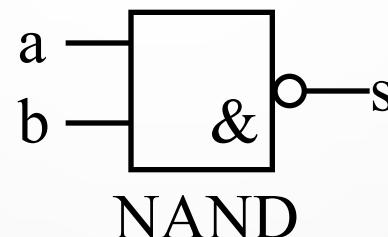
$$x = \overline{a \cdot b} = \overline{a} + \overline{b}$$

Table(au) de vérité

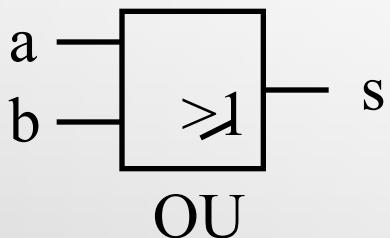
Représentation graphique



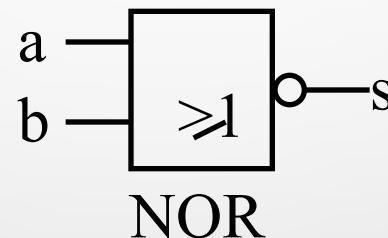
ET



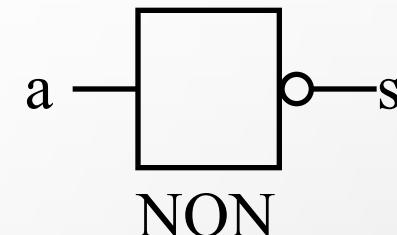
NAND



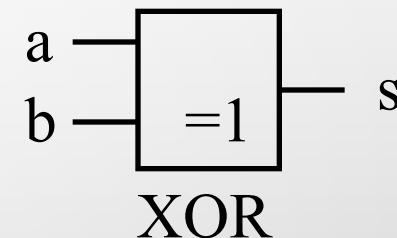
OU



NOR



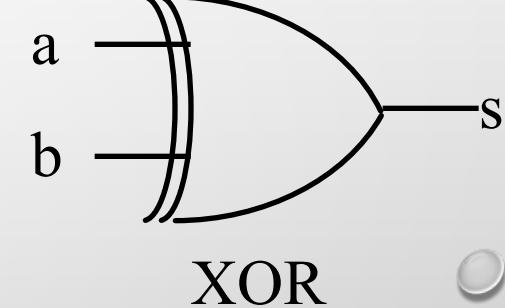
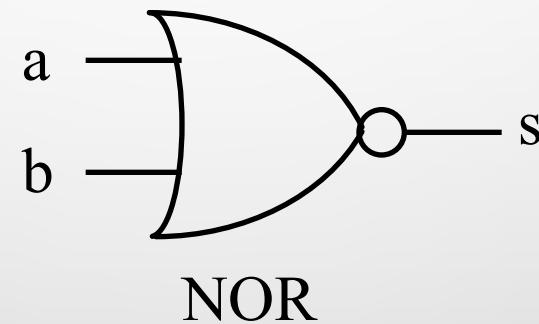
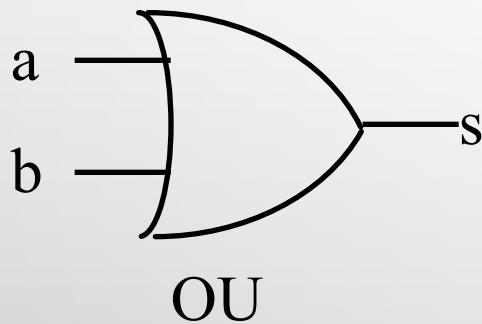
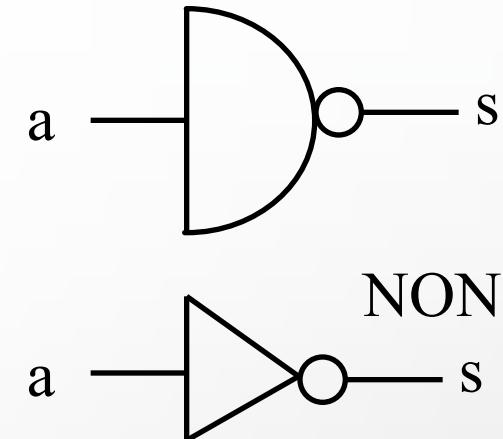
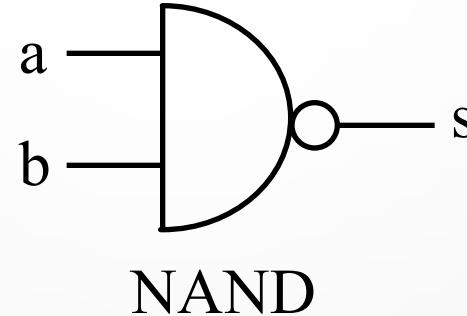
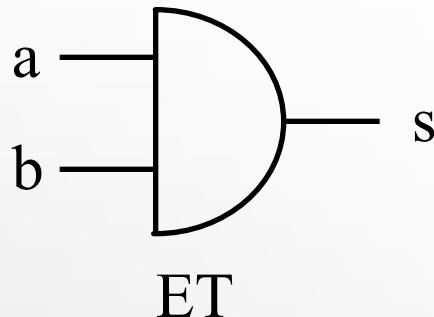
NON



XOR

Nb : La représentation française est utilisée en symboles logiques
Norme ANSI/IEEE Std: 91-1984 : Standard Graphic Symbols for Logic Functions

Représentation graphique



Nb : La norme américaine est la norme utilisée dans les « datasheets » pour représenter les diagrammes (schémas) logiques.

Fonctions NAND et NOR

- A l'aide du **théorème de De Morgan**, toute fonction logique peut s'écrire uniquement avec des portes NAND (resp. NOR)
- NAND (resp. NOR) est un **GOC** (complétude du NAND (NOR))
- Grandes performances technologiques
- Ou exclusif est également une fonction très importante

Méthodes de génération

Somme de produits + 2 complémentations → NAND

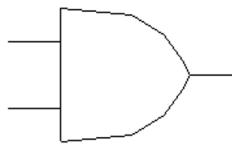
$$ab\bar{c} + a\bar{b}d + e = \overline{\overline{ab\bar{c}} + \overline{a\bar{b}d} + \overline{e}} = \overline{\overline{ab\bar{c}}}\overline{\overline{a\bar{b}d}}\overline{\overline{e}}$$

Produit de sommes + 2 complémentations → NOR

$$(a + \bar{b} + c)(a + \bar{d}) = \overline{(a + \bar{b} + c)(a + \bar{d})} = \overline{(a + \bar{b} + c)} + \overline{(a + \bar{d})}$$

Complétude du NAND

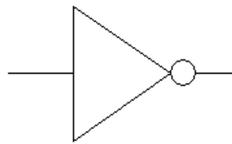
$$S = A \cdot B$$



ET

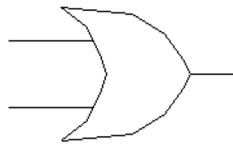
$$S = ((A \cdot B)')'$$

$$S = A'$$



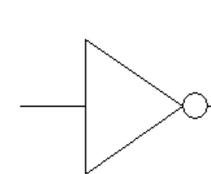
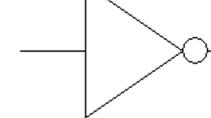
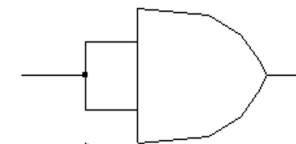
NON

$$S = A + B$$



OU

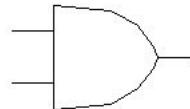
$$S = (A \cdot A)'$$



$$S = (A' \cdot B')' = A'' + B'' = A + B$$

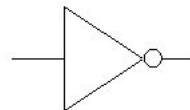
Complétude du NOR

$$S = A \cdot B$$



ET

$$S = A'$$

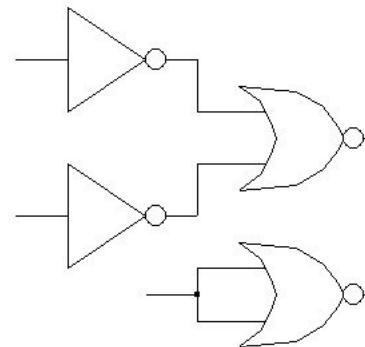


NON

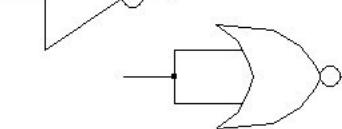
$$S = A + B$$



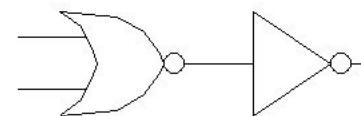
OU



$$S = (A' + B')' = A'' + B'' = A + B$$



$$S = (A + A)'$$



$$S = ((A + B)')' = A + B$$

• Les fonctions de n variables

• Fonction logique à n variables $f(a,b,c,d,\dots,n)$

$$[0,1]^n \rightarrow [0,1]$$

- Une fonction logique ne peut prendre que deux valeurs
- La combinaisons des variables d'entrées forment un ensemble fini (2^n)
- Descriptions, preuves possibles par énumération
 - Comparer $f(a,b,c,\dots,n)$ et $g(a,b,c,\dots,n)$
= comparer les tables représentant f et g

La table de fonction logique = table de vérité

Définition : (a,b,c,\dots,n) = vecteur d'entrée

Propriétés : conséquences

Fonctions Logiques à une variable a

■ 1 variable soit 4 fonctions possibles (2^{2^n}):

- $f(a) = 0$: fonction constante à 0
- $f(a) = 1$: fonction constante à 1
- $f(a) = a$: fonction identité (Buffer)
- $f(a) = \bar{a}$: fonction complément ou fonction NON

Fonction de 2 variables

- Fonctions Logiques de deux variables a et b
 - 2 variables soit 16 fonctions possibles 2^{2^n}

$f = a \cdot b$: fonction ET

$f = a + b$: fonction OU

$f = a \oplus b$: fonction OU-EXCLUSIF

$f = \overline{a \cdot b}$: fonction NON-ET

$f = \overline{a + b}$: fonction NON-OU

$f = \overline{a \oplus b}$: fonction NON-OU-EXCLUSIF

etc...

Les fonctions de n variables

- Fonction logique à n variables $f(a,b,c,d,\dots,n)$

$$[0,1]^n \rightarrow [0,1]$$

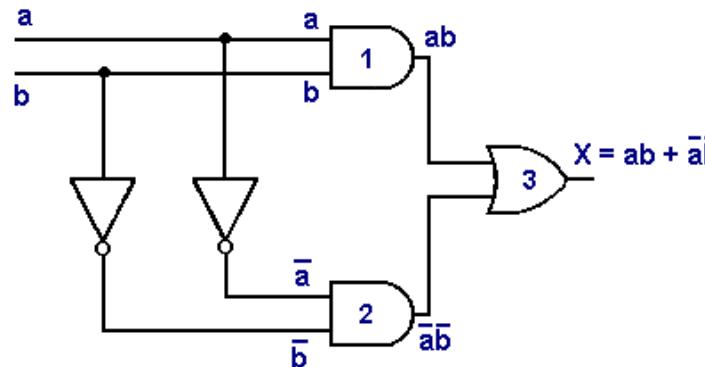
- n variables soit 2^{2^n} fonctions possibles

- 3 variables → 256 fonctions possibles
- 4 variables → 65536 fonctions possibles
- etc ...

Toute la difficulté réside dans la représentation quand le nombre de variables d'entrées devient important

• Représentation des fonctions

- Schéma symbolique
- Table de vérité
- Tableau de Karnaugh



a	b	x
0	0	1
0	1	0
1	0	0
1	1	1

Tableau + règles de symétries

- Équations logiques
- Chronogrammes : Graphe d'évolution temporelle



Attention plusieurs niveaux d'abstraction:
- analogique
- temps de réponse

Attention: multiniveaux

Certaines descriptions de fonctions numériques (datasheet) font appel à d'autres niveaux



Table 11-1 Input Characters and States

Character	State
D	Low
U	High
N	Unknown
Z	Tristate

Table 11-2 Output Characters and State

Character	State
L	Low
H	High
X	Unknown
T	Tristate

a	b	0	1	Z	X
0	0	X	0	X	
1	X	1	1	X	
Z	0	1	Z	X	
X	X	X	X	X	

Table de $s = a$ connecté à b

U non défini

X conflit +, W conflit-

Z haute impédance logique

L niveau bas haut

H niveau haut faible

Ensemble normalisé : MVL9 (U,X,0,1,Z,W,L,H)

Utilisé par tous les simulateurs numériques

Logique positive ou négative

0/1 sont représentées par des tensions, courants, fréquences etc.

Classiquement, on utilise des **TENSIONS**.

0 / 1 correspondent à deux niveaux :

Niveau haut = H (le plus positif)

Niveau bas = L (le plus négatif)

Plusieurs possibilités pour représenter l'information

Association d'une information binaire à un niveau :

Convention positive: Signal actif H 1

(ou logique positive) Signal inactif L 0



Convention négative: Signal inactif H 0

(ou logique négative) Signal actif L 1

La logique positive est plus « intuitive », la logique négative est utilisée dans certain cas, car elle permet d'optimiser la vitesse ou la consommation d'un circuit

Formes canoniques

f fonction logique de n variables

■ On appelle « **minterme** » de n variables, l'un des produits de ces variables ou de leurs complémentaires.

■ On appelle « **Maxterme** » de n variables, l'une des sommes de ces variables ou de leurs complémentaires.

Exemple $n = 4$ variables
 $A = \{a, b, c, d\}$

$m = a \cdot b \cdot c \cdot d$ est un minterme

$m = \bar{a} \cdot \bar{b} \cdot c \cdot d$ est un autre minterme

$m = a \cdot \bar{b} \cdot c$ n'est pas un minterme

$M = a + b + c + d$ est un maxterme

$M = \bar{a} + \bar{b} + c + d$ est un autre maxterme

$M = a + \bar{b} + c$ n'est pas un maxterme

Indexation et nombre de mintermes et maxtermes

- Pour chaque « m--terme », on construit un code binaire en posant 1 si une variable est présente, 0 si son complémentaire est présent.
- On convertit ce code binaire en base décimale pour obtenir l'indice du m—terme

$$m_i = a\bar{b}c\bar{d} \rightarrow (1010)_2 \rightarrow i = 10$$

- Si deux m--termes sont différents, leurs indices sont différents.
- Le nombre de m--terme de n variables vaut 2^n

Indexation et nombre de mintermes et maxtermes

- Le complémentaire d'un (max)minterme m est un (min)maxterme M .

$$\overline{m_i} = M_{2^n - 1 - i} \quad \text{et} \quad \overline{M_j} = m_{2^n - 1 - j}$$

- Exemple

$$m_i = a\overline{b}c\overline{d} \rightarrow (1010)_2 \rightarrow \text{Décimal : } i = 10$$

$$\overline{m_{10}} = \overline{a\overline{b}c\overline{d}} = \overline{a} + b + \overline{c} + d = M_5 \rightarrow (0101)_2 \rightarrow (2^n - 1 - i) \rightarrow 5$$

Indexation et nombre de mintermes et maxtermes

■ Théorème:

$$\sum_{i=0}^{2^n-1} m_i = 1 \quad \text{et} \quad \prod_{j=0}^{2^n-1} M_j = 0$$

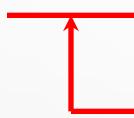
■ Soit f une expression booléenne écrite sous la forme d'une somme de mintermes (respectivement d'un produit de maxtermes), son complément f est la somme de tous les mintermes (respectivement le produit de tous les maxtermes) qui ne figurent pas dans f .

Formes canoniques

Une fonction est sous **forme canonique** (ou **normale**) si chaque terme contient toutes les variables. L'écriture sous forme canonique est unique.

Exemples :

$$f(x, y, z) = x \cdot y \cdot z + \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot z$$



Minterme ou intersection de base

Première forme canonique ou forme normale disjonctive

$$f(x, y, z) = (x + \bar{y} + z) \cdot (\bar{x} + y + \bar{z})$$



Maxterme ou réunion de base

Deuxième forme canonique ou forme normale conjonctive

Formes canoniques

Si la fonction n'est pas sous forme normale *i.e.* Une des variables (au moins) ne figure pas dans un des termes

→ La fonction est sous une forme simplifiée

$$f(a, b, c) = abc + \bar{a}b\bar{c} + a\bar{b}\bar{c}$$

Première forme canonique

$$f(a, b, c) = ab(c + \bar{c}) + \bar{a}b\bar{c}$$

Forme simplifiée

$$f(a, b, c) = b(a + \bar{a} \cdot \bar{c})$$

Forme simplifiée

$$f(a, b, c) = b(a + \bar{c})$$

Forme simplifiée

Premier théorème d'expansion de Shannon mintermes

$$F(a, b, c, \dots)$$

$$F(a, b, c, \dots) = a \cdot F(1, b, c, \dots) + \bar{a} \cdot F(0, b, c, \dots)$$

Si $a = 1$: $F(1, b, c, \dots) = 1 \cdot F(1, b, c, \dots) + 0 \cdot F(0, b, c, \dots)$

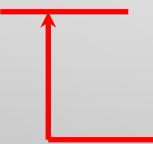
Si $a = 0$: $F(0, b, c, \dots) = 0 \cdot F(1, b, c, \dots) + 1 \cdot F(0, b, c, \dots)$

Pour 2 variables :

$$F(a, b) = a \cdot F(1, b) + \bar{a} \cdot F(0, b)$$

$$F(a, b) = a \cdot (b \cdot F(1, 1) + \bar{b} \cdot F(1, 0)) + \bar{a} \cdot (b \cdot F(0, 1) + \bar{b} \cdot F(0, 0))$$

$$F(a, b) = a \cdot b \cdot F(1, 1) + a \cdot \bar{b} \cdot F(1, 0) + \bar{a} \cdot b \cdot F(0, 1) + \bar{a} \cdot \bar{b} \cdot F(0, 0)$$



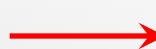
La fonction F vaut 0 ou 1

Premier théorème d'expansion de Shannon suite

$$F(a, b) = a \cdot b \cdot F(1,1) + a \cdot \bar{b} \cdot F(1,0) + \bar{a} \cdot b \cdot F(0,1) + \bar{a} \cdot \bar{b} \cdot F(0,0)$$

Pour chaque i et j , le point de la fonction $f(i,j)$ dépend du processus

A	b	F
0	0	$f(0,0)$
0	1	$f(0,1)$
1	0	$f(1,0)$
1	1	$f(1,1)$



$$F(a, b) = \bar{a} \cdot b + a \cdot \bar{b}$$

La première forme canonique ne laisse apparaître que les termes qui valent 1

Il y a 2^n mintermes possibles. La somme des 2^n mintermes valent 1. (Fonction valant 1 partout)

Premier théorème d'expansion de Shannon deuxième forme canonique : maxtermes

$$F(a, b, c, \dots)$$

$$F(a, b, c, \dots) = (a + F(0, b, c, \dots)).(\bar{a} + F(1, b, c, \dots))$$

Si $a=0$: $F(0, b, c, \dots) = (0 + F(0, b, c, \dots)).(1 + F(1, b, c, \dots))$

\uparrow neutre + \uparrow absorbant +
neutre .

Si $a=1$: $F(1, b, c, \dots) = (1 + F(0, b, c, \dots)).(0 + F(1, b, c, \dots))$

Pour deux variables :

$$\begin{aligned} F(a, b) &= (a + b + F(0,0)).(\bar{a} + b + F(1,0)). \\ &\quad (a + \bar{b} + F(0,1)).(\bar{a} + \bar{b} + F(1,1)) \end{aligned}$$

Premier théorème d'expansion de Shannon suite

$$F(a, b) = a \cdot b \cdot F(1,1) + a \cdot \bar{b} \cdot F(1,0) + \bar{a} \cdot b \cdot F(0,1) + \bar{a} \cdot \bar{b} \cdot F(0,0)$$

Maxtermes

$$F(a, b) = (a + b + F(0,0)) \cdot (\bar{a} + b + F(1,0)) \cdot (a + \bar{b} + F(0,1)) \cdot (\bar{a} + \bar{b} + F(1,1))$$

A	b	F
0	0	0 f(0,0)
0	1	1 f(0,1)
1	0	1 f(1,0)
1	1	0 f(1,1)

$$\longrightarrow F(a, b) = (a + b) \cdot (\bar{a} + \bar{b})$$

Que les termes Valant 0

Il y a 2^n maxtermes possibles. Le produit des 2^n maxtermes vaut 0.
(Fonction valant 0 partout)

Expression numérique de formes canoniques

Équivalent décimal	c Poids 4	b Poids 2	a Poids 1	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

$$F(c, b, a) = \bar{c}ba + c\bar{b}a + cb\bar{a} + cba = \sum_{(3,5,6,7)}$$

Simplification d'une fonction booléenne

- Simplification d'une fonction logique: mise en œuvre de méthodes permettant de décrire la même fonction sous une forme plus simple tout en conservant les caractéristiques de la fonction
- Attention que veut dire « plus simple » ?
- Forme d'optimisation (minimisation)
 - S'effectue toujours selon un ou plusieurs critères
 - Exemples pour circuits logiques
 - Moindre coût (production)
 - Circuits à temps de réponse le plus rapide
 - Consommation d'énergie minimale (systèmes embarqués)
- Souvent le minimisation n'est que formelle !

Minimisation d'une fonction booléenne

Soit à minimiser: $s = \overline{c}ba + c\overline{b}a + cb\overline{a} + cba$

$$s = \overline{c}ba + c\overline{b}a + cb\overline{a} + cba + cba + cba \Leftarrow (a + a = a)$$

$$s = \overline{c}ba + cba + c\overline{b}a + cba + cb\overline{a} + cba$$

$$s = (\overline{c} + c)ba + ca(\overline{b} + b) + cb(a + \overline{a})$$

$$s = ba + ca + cb$$

Minimisation d'une fonction booléenne principe de base

- X (x,y,z...) expression booléenne de n variables

$$s = Xa + X\bar{a} = X(a + \bar{a})$$



La difficulté est de mettre ces termes en évidence
→ Intérêt du diagramme de karnaugh

Diagramme de Karnaugh

- ▶ Diagramme de Karnaugh: grille comportant un nombre de « cases » égal au nombre de combinaisons des variables de la fonction booléenne qu'on se propose à simplifier
 - 3 variables => 2^3 cases = 8 cases
 - 4 variables => 2^4 cases = 16 cases
- ▶ Chaque case représente une combinaison booléenne des variables
- ▶ Lorsque l'on passe d'une case adjacente à une autre, seule une variable change d'état

Diagramme de Karnaugh (16 cases)

BA	00	01	11	10
DC	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

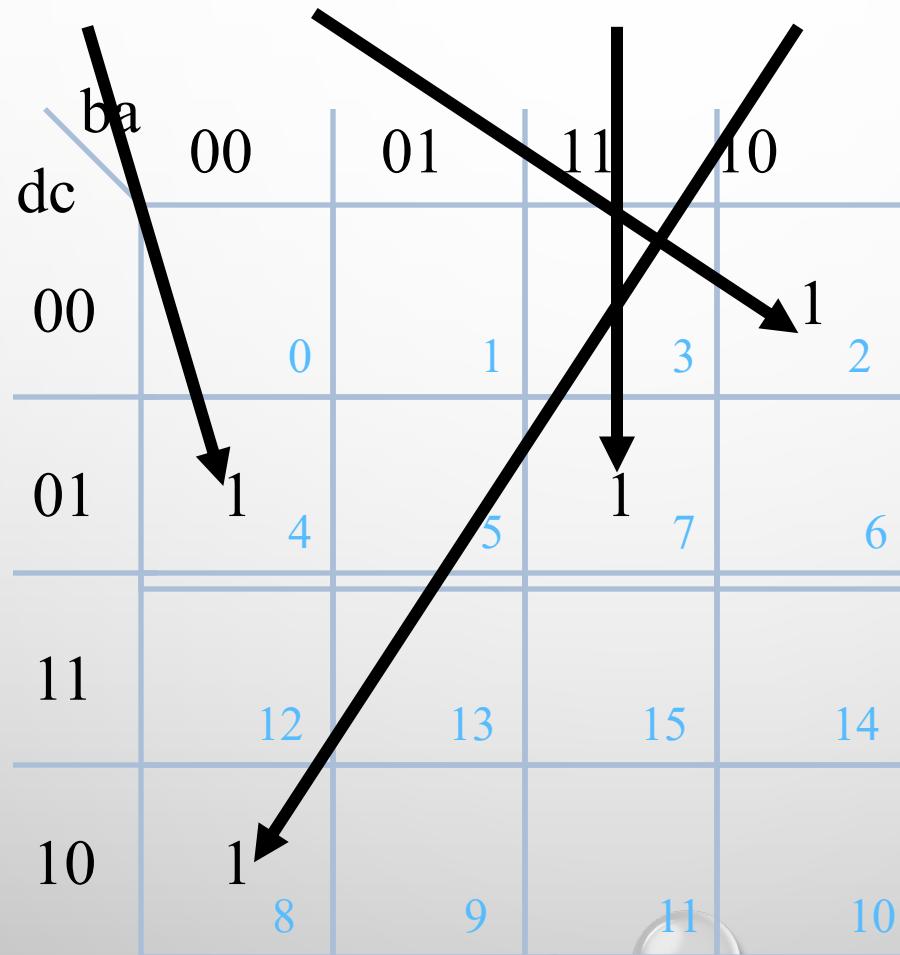
Code binaire réfléchi

en rouge l'équivalent décimal de la combinaison représentée par la case

Exemple de représentation

$$x = \overline{d}c\overline{b}\overline{a} + \overline{d}\overline{c}b\overline{a} + \overline{d}cba + d\overline{c}\overline{b}\overline{a}$$

$$X = (4, 2, 7, 8)$$



Apport du diagramme de Karnaugh

		ba	00	01	11	10
		c	0		1	
		0				
		1		1	1	1
		1				

$$x = \bar{c}ba + c\bar{b}a + cb\bar{a} + cba$$

Chaque terme correspond à des cases adjacentes

Apport du diagramme de Karnaugh

ba \ c	00	01	11	10
0			1	
1	1	1	1	

L'équation des regroupements est obtenue en fonction des variables qui ne changent pas d'état.

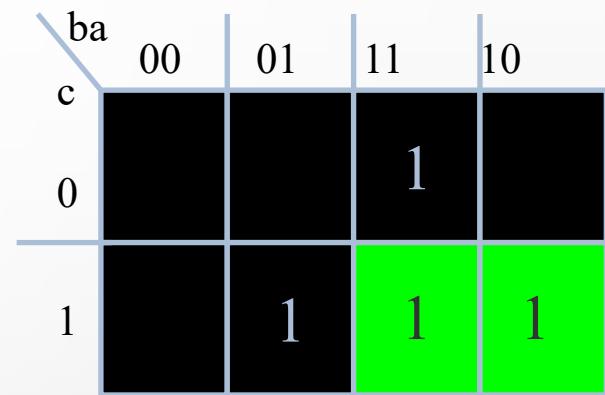
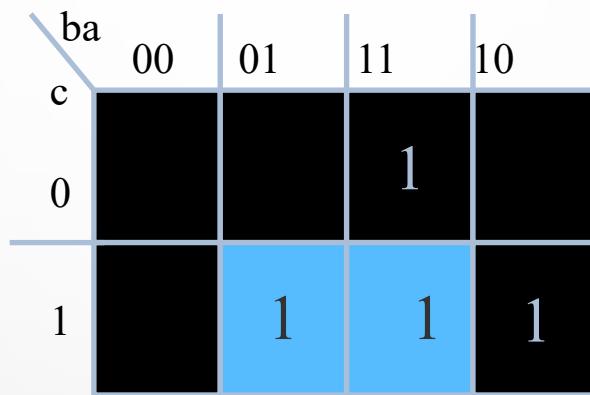
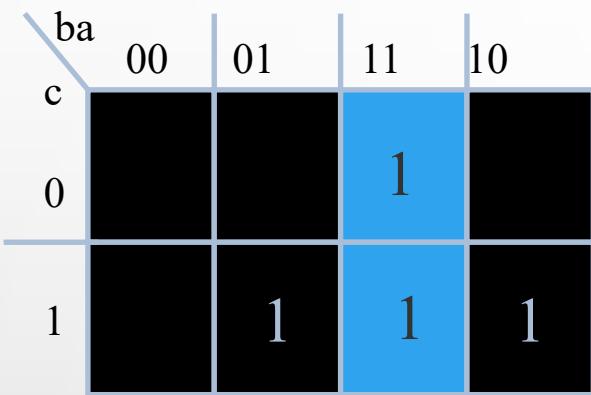
$$x = \overline{c}ba + c\overline{b}a + cb\overline{a} + cba$$

Les simplifications, à l'aide du tableau, se déduisent de regroupements graphique de cases.

→ Un regroupement correspond à un ensemble de cases adjacentes (le nombre de cases doit être une puissance de 2. Ex: 2, 4, 8, 16 cases ...Etc.). La forme des regroupements doit correspondre à des rectangles ou à des carrés.

→ Les regroupements les plus grands permettent d'obtenir les équations les plus simples.

Apport du diagramme de Karnaugh (2)



$$x = b.a + c.a + c.b$$

Apport du diagramme de Karnaugh (2)

Les tableaux de Karnaugh permettent d'éviter des problèmes d'aléas (états indésirables des sorties lors du changement de certaines variables d'entrées. Ces aléas se produisent pendant des durées très courtes, et sont dus aux temps de propagation des portes logiques).

Remarque 1

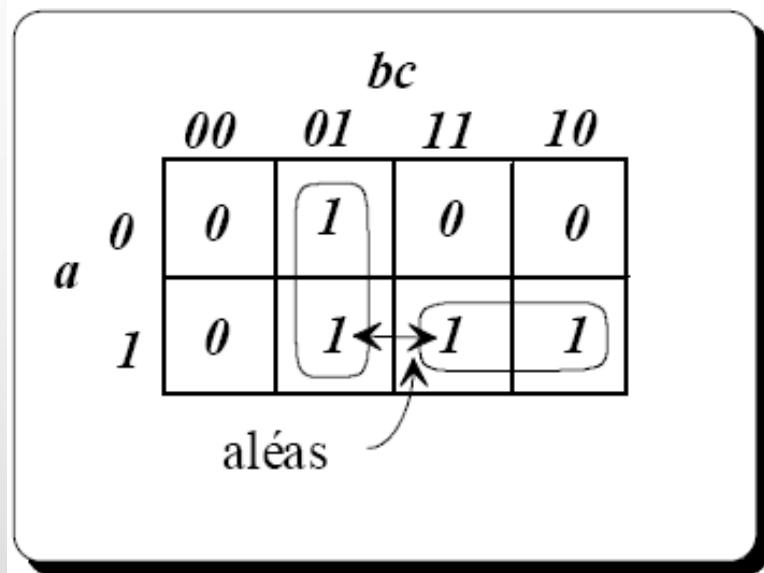
Les problèmes d'aléas sont souvent très difficiles à visualiser à l'oscilloscope, car leur durée (quelques nS) rend la visualisation parfois presque impossible

Remarque 2

Il n'est pas toujours indispensable de se préoccuper des aléas. Lorsqu'une variation brève d'une sortie ne peut pas entraîner un dysfonctionnement (ex: signal envoyé à un optocoupleur, pour un système de commande PWM). Il n'est pas gênant que l'affichage soit perturbé pendant un temps très court)

Appart du diagramme de Karnaugh (2)

Détection d'un aléas



$$f(a, b, c) = \bar{b}c + ab$$

Problème d'aléas, à chaque fois que 2 regroupements sont adjacents, et qu'il n'y a pas d'intersection.

Explication du phénomène

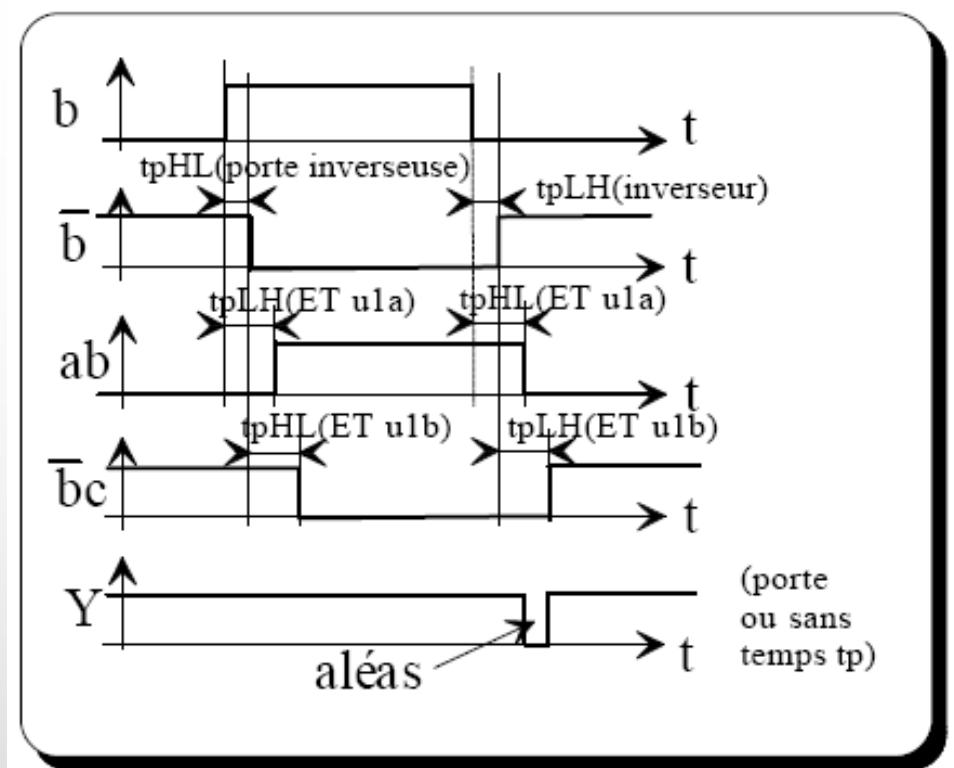


Diagramme temporel avec $a=c=1$.

Apport du diagramme de Karnaugh (2)

Il est possible de résoudre les problèmes d'aléas, directement à l'aide du tableau de Karnaugh. Pour cela il suffit d'ajouter un (ou des) regroupement(s) supplémentaire(s), afin de ne plus avoir de regroupements adjacents.

L'équation de f devient alors:

$$f(a, b, c) = \bar{b}c + ab + ac$$

Le dernier terme ac supprime l'aléas, car si $a=1$ et $c=1$, f est obligatoirement à 1, indépendamment de l'état ou des transitions de b .

		bc			
		00	01	11	10
a	0	0	1	0	0
	1	0	1	1	1

ajout d'un regroupement

Apport du diagramme de Karnaugh (2)

Regroupement de "0" avec les tableaux de Karnaugh

Il est parfois plus facile de travailler avec le complément de la sortie Y (lorsqu'il y a moins de 0, et/ou lorsque les regroupements de 0 sont plus simples). Dans le tableau ci-contre, il faudrait 3 (ou 4) regroupements (4 pour supprimer tout aléas) si on désire écrire l'équation de \bar{Y} . Alors que l'équation de Y ne nécessite que 2 regroupements.

$$\bar{Y} = \bar{f}(a, b, c) = \bar{b}\bar{c} + abc$$

On utilise alors les règles de bases pour obtenir Y (si besoin).

		bc			
		00	01	11	10
a	0	0	1	1	1
	1	0	1	0	1

$$Y = f(a, b, c) = \overline{\bar{b}\bar{c} + abc} = \overline{\bar{b}\bar{c}} \cdot \overline{abc} = (b + c) \cdot \overline{abc}$$

Il faut donc 1 porte OU, une porte ET et une porte NAND à 3 entrées.

Apport du diagramme de Karnaugh (2)

Inconvénient des tableaux de Karnaugh avec les fonctions OU exclusif

Les tableaux de Karnaugh sont plus adaptés dans le cas où les solutions les plus courtes utilisent des portes de type ET, OU, et des inverseurs.

Il est souvent difficile de voir les portes NAND, NOR, XOR ou XNOR directement sur le tableau.

→ Théorème De Morgan (Algèbre de Boole)

Exemple:

$$Y = a \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + a \cdot b \cdot c + \bar{a} \cdot b \cdot \bar{c}$$

$$Y = a \cdot (\bar{b} \cdot \bar{c} + b \cdot c) + \bar{a} \cdot (\bar{b} \cdot c + b \cdot \bar{c})$$

$$Y = a \oplus b \oplus c$$

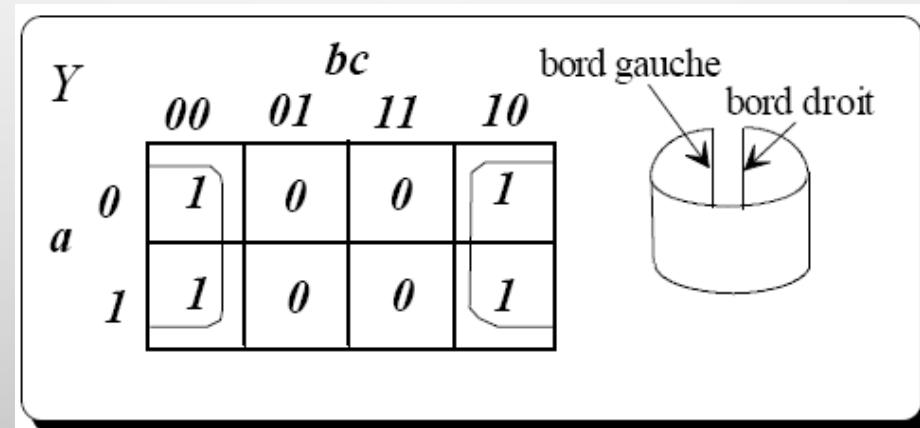
		bc				
		00	01	11	10	
a		0	0	1	0	1
		1	1	0	1	0

Apport du diagramme de Karnaugh (2)

Regroupements particuliers

Attention: le bord gauche du tableau rejoint le bord droit (comme si le tableau était enroulé sur un tube). De même le bord supérieur rejoint le bord inférieur). Ainsi dans l'exemple ci-dessous, l'équation de Y correspond à:

$$Y = f(a, b, c) = \bar{c}$$



Apport du diagramme de Karnaugh (2)

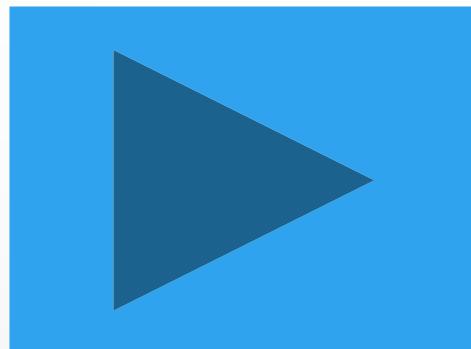
Tableau à 5 variables

Tableaux de Karnaugh
à 5 variables

Deux modes de
représentations

Tableau à 5 variables

Utilisation de programmes d'aide



Online Karnaugh map : <http://www.32x8.com/pos.html>

- ▶ Systèmes Numériques, Thomas L. Floyd, Ed Reynald et Goulet, Dunod, 2004.
- ▶ Circuits Numériques, Ronald J. Tocci, Ed Reynald et Goulet, Dunod, 1996
- ▶ Electronique numérique 1 *Circuits logiques combinatoires*, Tertulien Ndjountche, Collection Electronique dirigée par Robert Baptist, Edition ISTE, 2016
- ▶ Electronique numérique 2 *Circuits logiques séquentiels et arithmétiques*, Tertulien Ndjountche, Collection Electronique dirigée par Robert Baptist, Edition ISTE, 2016