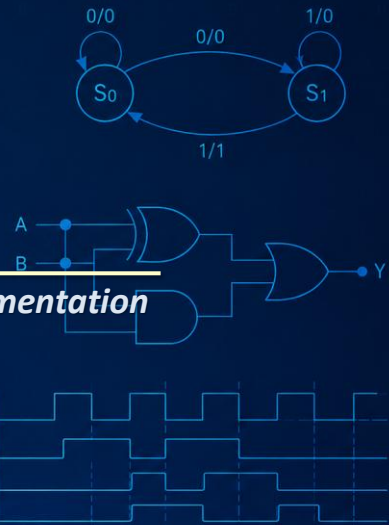


III. Logique combinatoire

Algèbre de Boole et fonctions booléennes, implémentation


ENSGTI
 ÉCOLE D'INGÉNIEURS

Arnaud Martin

1

1


 III. Logique
combinatoire

Back to basics

développée de longue date

 de λόγος / lógos signifiant
raison, langage et raisonnement

- Conceptuellement :
 - En philosophie (le vrai, le faux) : étude des règles formelles que doit respecter toute argumentation correcte. III^e siècle av. J.-C.
 - En mathématique : algèbre de Boole. XIX^e siècle.



Steve Mould, Water computing



Pascaline, 1642

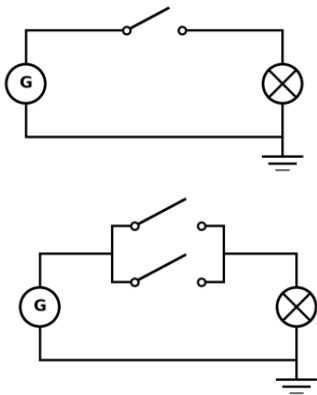
- En pratique :
 - [Hydraulique](#)
 - Mécanique
 - [Électrique](#) : relais-interrupteurs (Model K 1937) → tubes à vide (ENIAC 1945)

→ diodes (SEAC 1950) → transistors (Manchester Transistor Computer 1953)

2

2

- Électricité : logique d'interrupteur



3

3

- Algèbre de Boole définie sur E_2 constitué des éléments $\{0,1\}$

- Trois opérations de base :

complémentation		union + max OU			intersection · min ET		
a	$\bar{a}$	a	b	s	a	b	s
0	1	0	0	0	0	0	0
1	0	0	1	1	0	1	0
		1	0	1	1	0	0
		1	1	1	1	1	1

- 0 : élément neutre de l'union, élément minimum.
- 1 : élément neutre de l'intersection, élément maximum.

4

4

$$\forall a, b, c \in E_2$$

- complémentation : $a. \bar{a} = 0$
 $a + \bar{a} = 1$
- commutativité : $a.b = b.a$
 $a + b = b + a$
- associativité : $(a.b).c = a.(b.c)$
 $(a + b) + c = a + (b + c)$
- distributivité : $a.(b + c) = a.b + a.c$
 $a + (b.c) = (a + b).(a + c)$

5

5

$$\forall a, b, c \in E_2$$

- idempotence : $a.a = a$
 $a + a = a$
- absorption : $a + a.b = a$
 $a.(a + b) = a$
- involution : $\bar{\bar{a}} = a$
- Théorème de Morgan : $\overline{a + b} = \bar{a}. \bar{b}$; $\overline{a.b} = \bar{a} + \bar{b}$
- Théorème de Shannon : $a + b = a.b + a.\bar{b} + \bar{a}.b$
- Théorème du Consensus : $a.b + \bar{b}.c = a.b + \bar{b}.c + \boxed{a.c}$
 $(a + b).(\bar{b} + c) = (a + b).(\bar{b} + c).\boxed{(a + c)}$

à voir par
vous même

Termes de
Consensus

b variable biforme

6

6

- repérer les variables biformes,
- introduire le terme de "Consensus" s'il permet de réduire l'expression par absorption,
- absorption,
- retirer le terme de "consensus".

Exemple :

$$F = f\bar{e}b + hgf\bar{e}dca + hg\bar{b}$$

ajout terme Consensus

$$F = f\bar{e}b + hgf\bar{e}dca + hg\bar{b} + hgf\bar{e}$$

Absorption

retrait Consensus

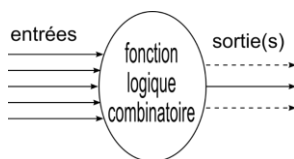
$$F = f\bar{e}b + hg\bar{b}$$



7

7

$E_2 \times E_2 \times \dots \times E_2$ dans E_2 .



entrées		sortie(s)
a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité

Objectif :

Exprimer la ou les sorties en fonction des entrées (puis y associer des portes logiques – support matériel : synthèse logique)

Deux façons d'exprimer une fonction booléenne :

- la forme **somme de produit** ou disjonctive normale $\Sigma\Pi$
- la forme **produit de somme** ou conjonctive normale $\Pi\Sigma$

Idée : avoir l'expression la plus compacte possible



Moins de support physique, moins de consommation, plus rapide

8

8

- Première forme canonique
 - On associe une variable binaire m_i à chaque entrée de la table
 - *minterm* , terme produit

m_0 est associé à la ligne 0, actif/vrai (à 1) pour la ligne 0

$m_0 = 1$ si $a = 0$ et $b=0$ c'est-à-dire si $\bar{a} = 1$ et $\bar{b} = 1$

$$m_0 = \bar{a} . \bar{b}$$

m_i	a	b	s
m_0	0	0	s_0
m_1	0	1	s_1
m_2	1	0	s_2
m_3	1	1	s_3

$$m_0 = \bar{a} . \bar{b} ; m_1 = \bar{a} . b ; m_2 = a . \bar{b} ; m_3 = a . b$$

$$s = s_0 m_0 + s_1 m_1 + s_2 m_2 + s_3 m_3$$

9

9

- Simplification de l'expression
 - minterms pour lesquels la fonction vaut 1

- Dans le cas de la fonction XOR

$$s = m_1 + m_2$$

$$s = \bar{a} . b + a . \bar{b}$$

m_i	a	b	s
m_0	0	0	0 (s_0)
m_1	0	1	1 (s_1)
m_2	1	0	1 (s_2)
m_3	1	1	0 (s_3)



Remarque : Réalisation simple avec des portes NAND

10

10

- Seconde forme canonique

- On associe une variable binaire M_i à chaque entrée de la table

- maxterm*, terme somme

- Exemple pour la fonction XOR

a	b	M_0	M_1	M_2	M_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

$$M_0 = a + b; M_1 = a + \bar{b}; M_2 = \bar{a} + b; M_3 = \bar{a} + \bar{b}$$

$$s = (s_0 + M_0) \cdot (s_1 + M_1) \cdot (s_2 + M_2) \cdot (s_3 + M_3)$$

M_i	a	b	s
M_0	0	0	0 (s_0)
M_1	0	1	1 (s_1)
M_2	1	0	1 (s_2)
M_3	1	1	0 (s_3)

Maxterm constitué de l'union (OU) de toutes les variables d'entrée, non complémentées si leur valeur est 0, complémentées si leur valeur est 1.

à voir par
vous même

11

11

- Simplification de l'expression

- On conserve les maxterm pour lesquels la fonction vaut 0

a	b	M_0	M_1	M_2	M_3	$M_0 M_3$	s
0	0	0	1	1	1	0	0
0	1	1	0	1	1	1	1
1	0	1	1	0	1	1	1
1	1	1	1	1	0	0	0

$$s = 0 \text{ si } M_0 = 0 \text{ ou si } M_3 = 0 \text{ c'est-à-dire si } M_0 \cdot M_3 = 0$$

$$s = M_0 \cdot M_3 \quad s = (a + b) \cdot (\bar{a} + \bar{b})$$



Remarque : Réalisation simple avec des portes NOR

à voir par
vous même

12

12

III. Logique combinatoire

Fonctions logiques

↓

Portes logiques

fonction	symbole	(ancien symbole)	table de vérité															
NON, NOT, inversion			<table border="1"> <tr><th>a</th><th>s</th></tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </table>	a	s	0	1	1	0									
a	s																	
0	1																	
1	0																	
ET, AND, intersection, min			<table border="1"> <tr><th>a</th><th>b</th><th>s</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	a	b	s	0	0	0	0	1	0	1	0	0	1	1	1
a	b	s																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OU, OR, union, max			<table border="1"> <tr><th>a</th><th>b</th><th>s</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	a	b	s	0	0	0	0	1	1	1	0	1	1	1	1
a	b	s																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NAND, non-ET			<table border="1"> <tr><th>a</th><th>b</th><th>s</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	a	b	s	0	0	1	0	1	1	1	0	1	1	1	0
a	b	s																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR, non-OU			<table border="1"> <tr><th>a</th><th>b</th><th>s</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	a	b	s	0	0	1	0	1	0	1	0	0	1	1	0
a	b	s																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR, ou-exclusif			<table border="1"> <tr><th>a</th><th>b</th><th>s</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	a	b	s	0	0	0	0	1	1	1	0	1	1	1	0
a	b	s																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

Aux fonctions logiques booléennes élémentaires sont associées des portes logiques.

Ces portes logiques ont une existence matérielle qui permet de réaliser concrètement les fonctions logiques.

Elles peuvent se présenter sous différentes formes:

- discrètes,
- intégrées,
- assemblées dans des blocs élémentaires,
- sous forme de matrice connectables ,
- ...

Idée : passer d'une fonction sur papier ou écran de PC à un support physique

III. Logique combinatoire

Fonctions logiques

↓

Portes logiques

Remarque 1 : en pratique, les équations de l'algèbre de Boole ne sont pas toujours vérifiées.

Idée : différence entre idéal (fonction logique) et implémentation (réalisation physique)

important !

Philips Semiconductors Product specification

Quadruple exclusive-OR gate HEF4030B gates

	V _{DD} V	SYMBOL	TYP.	MAX.	TYPICAL EXTRAPOLATION FORMULA
Propagation delays $I_H \rightarrow O_L$	5	t_{PHL}	85	175 ns	57 ns + (0,55 ns/pF) C _L
	10		35	75 ns	24 ns + (0,23 ns/pF) C _L
	15		30	55 ns	22 ns + (0,16 ns/pF) C _L
LOW to HIGH	5	t_{PLH}	75	150 ns	47 ns + (0,55 ns/pF) C _L
	10		30	65 ns	19 ns + (0,23 ns/pF) C _L
	15		25	50 ns	17 ns + (0,16 ns/pF) C _L
Output transition times HIGH to LOW	5	t_{FHL}	60	120 ns	10 ns + (1,0 ns/pF) C _L
	10		30	60 ns	9 ns + (0,42 ns/pF) C _L
	15		20	40 ns	6 ns + (0,28 ns/pF) C _L
LOW to HIGH	5	t_{FLH}	60	120 ns	10 ns + (1,0 ns/pF) C _L
	10		30	60 ns	9 ns + (0,42 ns/pF) C _L
	15		20	40 ns	6 ns + (0,28 ns/pF) C _L

14
14

III. Logique combinatoire

- 7 -



III. Logique combinatoire

Fonctions logiques

Portes logiques

Synthèse combinatoire

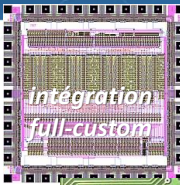
Synthèse combinatoire

fonction logique à réaliser sous forme de table ou d'équation

choix d'une cible technologique

réalisation matérielle de la fonction

- logique anarchique
- opérateurs élémentaires
- logique structurée (mémoire ROM, PAL, CPLD, FPGA)







III. Logique combinatoire

Fonctions logiques

Portes logiques

Synthèse combinatoire

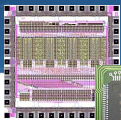
Synthèse combinatoire

fonction logique à réaliser sous forme de table ou d'équation

choix d'une cible technologique

réalisation matérielle de la fonction

- logique anarchique
- opérateurs élémentaires
- logique structurée (mémoire ROM, PAL, CPLD, FPGA)

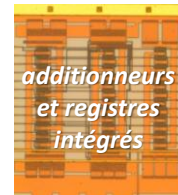
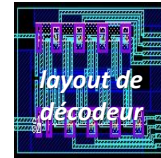



Note

Le choix de la cible technologique dépend

- de la complexité de la fonction à réaliser
- du coût
- de la vitesse de traitement nécessaire
- du savoir-faire du concepteur
- ...

- À base de porte logique AND, OR, NAND, NOR, XOR, INV.
 - méthode historique
 - adaptée à de très faibles complexités dans le cas de composants discrets
 - utilisation de composants discrets en voie d'obsolescence
- base pour les circuits intégrés full-custom
 - (ASIC: Application Specific Integrated Circuit)
- création de bloc élémentaire pour des structures plus complexes
 - cas des opérateurs arithmétiques
- Première étape de la synthèse : simplification des équations
 - algèbre de Boole
 - table de Karnaugh
 - logiciel de synthèse



17

17

- Outil graphique pour la simplification des équations
- Fait apparaître la symétrie sur les variables

symétrie b

ATTENTION !

c \ ba	0 0	0 1	1 1	1 0
0	s_0	s_1	s_3	s_2
1	s_4	s_5	s_7	s_6

symétrie a symétrie a

Si l'on fait un groupement autour de cet axe de symétrie, cela élimine la variable c

symétrie c

- Le groupement d'éléments se traduit par la suppression de variables dans l'expression de la fonction.

Groupe de 2 éléments : suppression d'une variable

Groupe de 4 éléments : suppression de deux variables

18

18

▪ Règles de simplification

- rechercher les « 1 » isolés,
- faire croître par adjacence les regroupements,
- faire des regroupements de taille maximale,
- ne prendre que les regroupements ou termes produits nécessaires pour prendre au moins une fois chaque 1 sans redondance.



- Quand la valeur de sortie n'est pas définie, on associe à la case de la table le caractère $\emptyset$ (don't care) ou X.

▪ Limité aux fonctions ayant au plus 4 entrées.

- Permet de créer des sous-structures élémentaires simples, puis cascadées. C'est le cas pour les opérateurs arithmétiques

19

19

Écrivez les équations booléennes d'une fonction active, pour les nombres premiers entre 0 et 7.

20

20

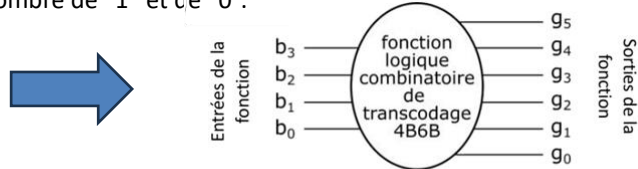
Synthèse combinatoire – 1- Logique anarchique - Table de Karnaugh

Exemple du transcodeur 4B6B

Le transcodage 4B6B étend un code de 4 bits en un code de 6 bits, contenant le même nombre de "1" et de "0".

	b ₃	b ₂	b ₁	b ₀	g ₅	g ₄	g ₃	g ₂	g ₁	g ₀
0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	1	0	0	1	1	0	1
2	0	0	1	0	0	1	0	0	1	1
3	0	0	1	1	0	1	0	1	1	0
4	0	1	0	0	0	1	0	1	0	1
5	0	1	0	1	1	0	0	0	1	1
6	0	1	1	0	1	0	0	1	1	0
7	0	1	1	1	1	0	0	1	0	1
8	1	0	0	0	0	1	1	0	0	1
9	1	0	0	1	0	1	1	0	1	0
A	1	0	1	0	0	1	1	1	0	0
B	1	0	1	1	1	1	0	0	0	1
C	1	1	0	0	1	1	0	0	0	1
D	1	1	0	1	1	0	1	0	0	1
E	1	1	1	0	1	0	1	0	1	0
F	1	1	1	1	1	0	1	1	0	0

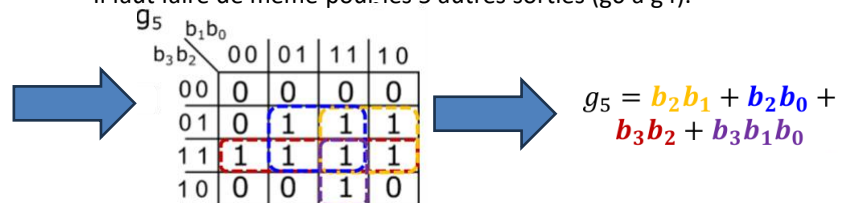
Table de vérité de la fonction logique de transcodage 4B6B



Fonction logique combinatoire à 4 entrées et 6 sorties.

Exemple de résolution pour la sortie g₅.

Il faut faire de même pour les 5 autres sorties (g₀ à g₄).

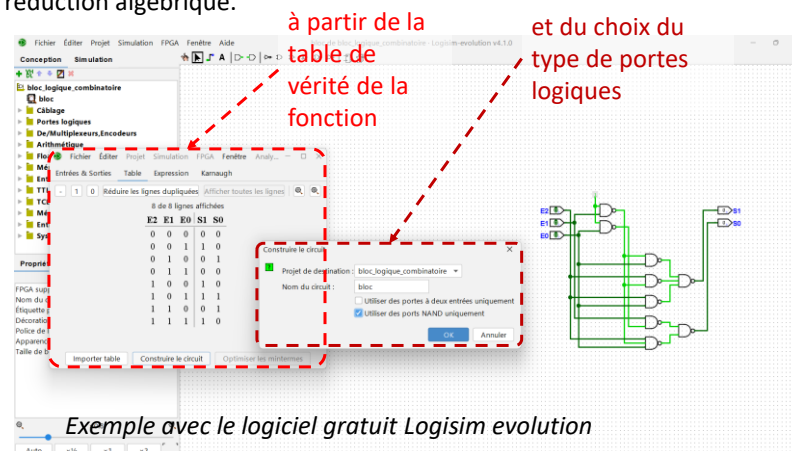


21

21

Synthèse combinatoire – 1- Logique anarchique – Logiciel de synthèse logique

- Algorithme de réduction algébrique.



Exemple avec le logiciel gratuit Logisim evolution

Logisim evolution: [Download](#)

22

22

Exemple de l'algorithme de Quine-Mac Cluskey (1956).

Utilise la 1^{ère} forme canonique ($\Sigma\P$).

Les combinaisons sont rassemblées en groupes, en fonction du nombre de 1 des minterms pour lesquels la fonction est vraie. Des simplifications sont faites entre les minterms adjacents de groupes consécutifs.

Sur un exemple :

d	c	b	a	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

rassemblement
des minterms

grp	N°	d	c	b	a
0	0	0	0	0	0
1	1	0	0	0	1
	2	0	1	0	0
2	3	0	1	0	1
	4	1	0	0	1
	5	1	0	1	0
3	6	0	1	1	1
	7	1	1	0	1
4	8	1	1	1	1

23

23

grp	N°	d	c	b	a
0	0	0	0	0	0
1	1	0	0	0	1
	2	0	1	0	0
2	3	0	1	0	1
	4	1	0	0	1
	5	1	0	1	0
3	6	0	1	1	1
	7	1	1	0	1
4	8	1	1	1	1

simplifications

grp	N°	d	c	b	a
0-1	0-1	0	0	0	X
	0-2	0	X	0	0
1-2	1-4	X	0	0	1
	2-3	0	1	0	X
	3-6	0	1	X	1
2-3	3-7	X	1	0	1
	4-7	1	X	0	1
	6-8	X	1	1	1
3-4	7-8	1	1	X	1

le N°5 n'est pas représenté

comparaison du minterm du groupe 0
avec les minterms du groupe 1

0-1	0	0	0	X
0-2	0	X	0	0

comparaison des minterms du groupe 1
avec les minterms du groupe 2

1-4	X	0	0	1
2-3	0	1	0	X

comparaison des minterms du groupe 2
avec les minterms du groupe 3

3-6	0	1	X	1
3-7	X	1	0	1
4-7	1	X	0	1

comparaison des minterms du groupe 3
avec le minterm du groupe 4

6-8	X	1	1	1
7-8	1	1	X	1

24

24

grp	N°	d	c	b	a
0-1	0-1	0	0	0	X
	0-2	0	X	0	0
1-2	1-4	X	0	0	1
	2-3	0	1	0	X
2-3	3-6	0	1	X	1
	3-7	X	1	0	1
	4-7	1	X	0	1
3-4	6-8	X	1	1	1
	7-8	1	1	X	1



Simplifications

grp	N°	d	c	b	a
0-1 ; 1-2	0-1 ; 2-3	0	X	0	X
1-2 ; 2-3	1-4 ; 3-7	X	X	0	1
2-3 ; 3-4	3-6 ; 7-8	X	1	X	1
	3-7 ; 6-8	X	1	X	1

Le N°5 n'est pas représenté

le N°5 n'est pas représenté

Comparaison du minterm du groupe 0-1
avec les minterms du groupe 1-2

0-1 ; 2-3 0 X 0 X

Comparaison des minterms du groupe 1-2
avec les minterms du groupe 2-3

1-4 ; 3-7 X X 0 1

Comparaison des minterms du groupe 2-3
avec les minterms du groupe 3-4

3-6 ; 7-8 X 1 X 1

3-7 ; 6-8 X 1 X 1

25

25

grp	N°	d	c	b	a
0-1 ; 1-2	0-1 ; 2-3	0	X	0	X
1-2 ; 2-3	1-4 ; 3-7	X	X	0	1
2-3 ; 3-4	3-6 ; 7-8	X	1	X	1
	3-7 ; 6-8	X	1	X	1

le N°5 n'est pas représenté

1 ^{er} réarrangement					
grp	N°	d	c	b	a
0	0	0	0	0	0
1	1	0	0	0	1
	2	0	1	0	0
2	3	0	1	0	1
	4	1	0	0	1
	5	1	0	1	0
3	6	0	1	1	1
	7	1	1	0	1
4	8	1	1	1	1

Finalement, la fonction est représentée par : (5), (0-1 ; 2-3), (1-4 ; 3-7), (3-6 ; 7-8)

ou encore : $d\bar{c}b\bar{a} + \bar{d}\bar{b} + \bar{b}a + ca$

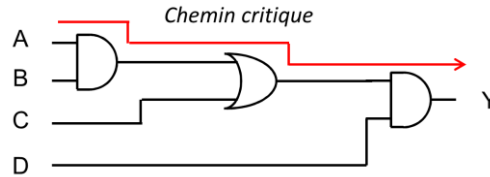
26

26

Chemin critique

- Le parcours qui va générer le délai de propagation le plus long entre les entrées et la sortie du circuit est a priori le chemin qui traverse le plus de portes logiques.

porte	Délai (ns)
NOT	30
AND	60
OR	70



Le chemin critique détermine le temps de calcul des opérateurs arithmétiques pour le calcul.

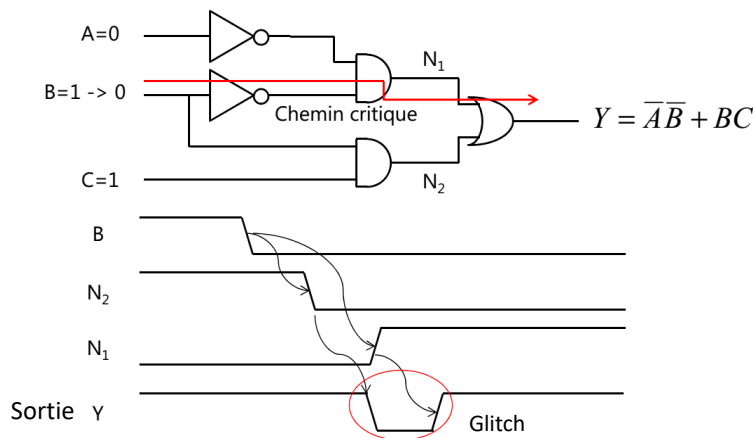


Le chemin critique limite la fréquence maximum de fonctionnement des systèmes synchrones.

27

27

Glitch : Variations transitoires des niveaux logiques avant d'atteindre un état.

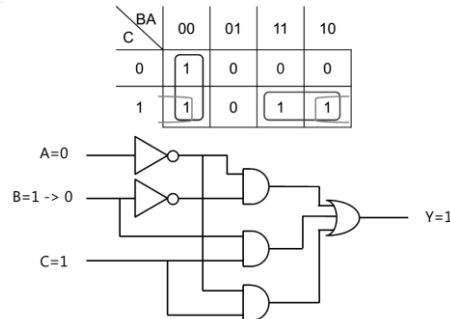


28

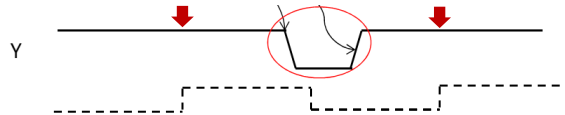
28

Glitch : Solutions

1. Les éliminer en modifiant le circuit
(par exemple termes redondants de Consensus)



2. Ne pas s'en préoccuper en utilisant une **conception synchrone** avec un cadencement d'horloge adapté (on ne prend en compte le signal que périodiquement, une fois stabilisé).



29

29

Sortie d'un circuit logique

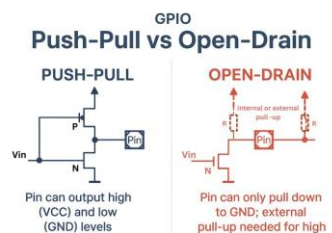
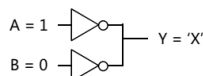
Types de sortie (étages de sortie) : la sortie d'un circuit logique peut être fournie soit par 2 transistors complémentaires « push-pull » ou par un seul transistor « drain ouvert ».

La sortie push-pull est pratique pour cascader des circuits de même technologie, même alimentation.

La sortie drain ouvert est pratique pour changer de niveau de tension ou la réalisation d'un ET câblé (bus i2c).

Sortance : capacité d'une porte logique à servir de source à d'autres portes logiques. Courant de sortie > somme de courant d'entrée des portes en aval. En CMOS limite de vitesse de fonctionnement.

Court-circuit (état indéterminé)



30

30

Opérateurs élémentaires

- multiplexeurs,
- démultiplexeur,
- décodeur,
- encodeur

Rmq

Pédagogique, en pratique plus utilisé

Rmq

Opérateurs élémentaires eux-mêmes constitués de portes logiques ET, NON, OU...

Rmq

Comme la logique anarchique avec des portes ET, OU, NON, ... discrètes, ce type d'implémentation est en voie d'obsolescence et réservé à des fonctions logiques simples

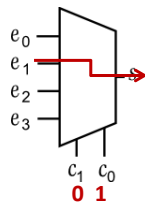
Idée : Les opérateurs ne sont plus utilisés pour la synthèse logique mais sont des éléments de base des systèmes numériques

31

31

Multiplexeur

- **Fonction d'aiguillage**
- Exemple d'un multiplexeur 4 vers 1



c ₁	c ₀	s
0	0	e ₀
0	1	e ₁
1	0	e ₂
1	1	e ₃

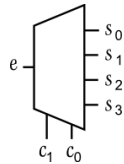
$$s = e_0 \cdot \bar{c}_1 \cdot \bar{c}_0 + e_1 \cdot \bar{c}_1 \cdot c_0 + e_2 \cdot c_1 \cdot \bar{c}_0 + e_3 \cdot c_1 \cdot c_0$$

32

32

Démultiplexeur

- Opérateur dual



$$\begin{aligned} s_0 &= \bar{c}_1 \cdot \bar{c}_0 \cdot e \\ s_1 &= \bar{c}_1 \cdot c_0 \cdot e \\ s_2 &= c_1 \cdot \bar{c}_0 \cdot e \\ s_3 &= c_1 \cdot c_0 \cdot e \end{aligned}$$

c ₁	c ₀	s ₀	s ₁	s ₂	s ₃
0	0	e	0	0	0
0	1	0	e	0	0
1	0	0	0	e	0
1	1	0	0	0	e

33

33

Décodeur

- Active une sortie parmi 2^N grâce à N signaux de commande.
- Utilisé pour « adresser ».
- Équivalent du démultiplexeur avec une entrée de donnée à « 1 »

- Exemple de décodeur 4 sorties

$$s_0 = \bar{c}_1 \cdot \bar{c}_0$$

$$s_1 = \bar{c}_1 \cdot c_0$$

$$s_2 = c_1 \cdot \bar{c}_0$$

$$s_3 = c_1 \cdot c_0$$

c ₁	c ₀	s ₀	s ₁	s ₂	s ₃
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Dans certains décodeurs discrets, la sortie active peut être au niveau bas et les inactives au niveau haut.

34

34

Logique structurée

- Structures universelles prédéfinies sous forme de matrice
- Forte densité d'intégration → VLSI
- Structure dépend du composant :
 - ROM, espace mémoire et système d'adressage
 - PAL, matrices de ET et de OU pouvant être interconnectées
 - CPLD, macro-cellules logiques interconnectables
 - FPGA

Note

structure la plus universelle

Note

Toutes les fonctions logiques ne sont pas réalisables, mais il y a de la marge

35

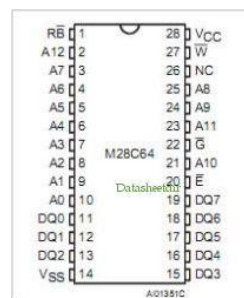
35

- Mémoire $M \times n$: sauvegarde de M mots de n bits.

- Exemple :

MC28C64C : 64 kbit ; 8kx8bits, parallel EEPROM

nombre de bits d'adresse ?



Idee : Prétexte pour parler des mémoires, n'est plus utilisé en pratique pour la synthèse logique, mais reste un élément fondamental de l'électronique numérique.

36

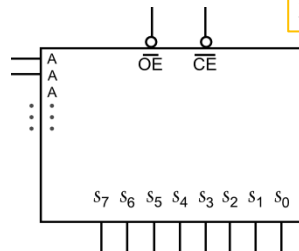
36

- La taille du mot mémorisé permet de définir autant de fonctions logiques que son nombre de bits.
- Le nombre de bits d'adresse correspond au nombre de variables des fonctions logiques.

Rmq

- Exemple** : implémentation de 5 fonctions logiques à 3 entrées

c	b	a	F ₄	F ₃	F ₂	F ₁	F ₀
0	0	0	0	1	1	0	0
0	0	1	1	1	0	1	0
0	1	0	1	0	0	0	1
0	1	1	0	0	1	0	0
1	0	0	0	1	0	1	0
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	1
1	1	1	0	0	1	1	1



Remarque :

plus petites mémoires
4k RAM 0,71€ 47C04 (2026)
2k EEPROM : 0,47€ 24C02 (2026)



37

37

historique (années 80)

Programmable Logic Array (82S100)

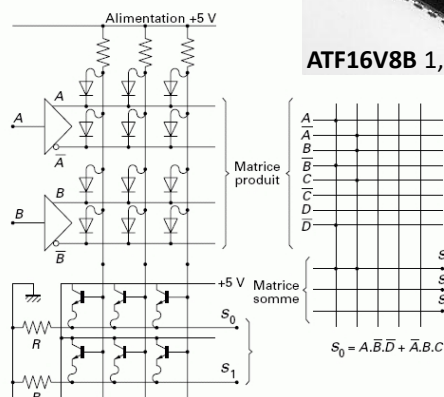
- Constituée de deux sous-ensembles :
 - une matrice de ET configurable,
 - une matrice de OU configurable.
- remplacé par des **PAL** (OU fixe)

ATF16V8

nb variables

nb sorties

Fabricants : Microchip, Lattice Semiconductor, AMD (Advanced Micro Devices), Texas Instrument

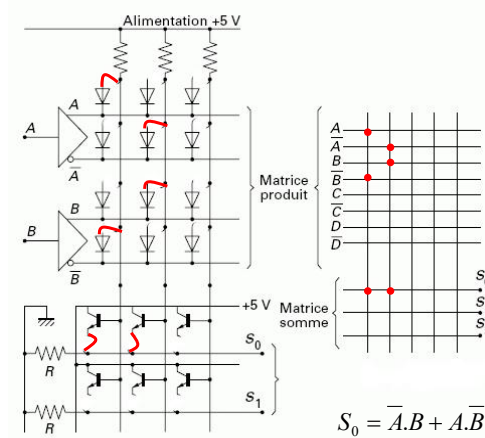


ATF16V8B 1,33 € (2026)

38

38

- implémentation de la fonction XOR dans un PLA :

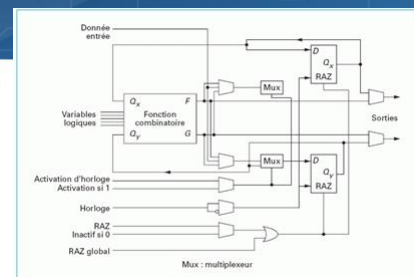


39

39

CPLD : Complex Programmable Logic Device
FPGA : Field Programmable Gate Array

- Réseaux logiques programmables complexes formés de blocs logiques disposés sous forme matricielle



bloc logique de FPGA
Look-Up Table (LUT)

Idée : composants de l'électronique numérique programmée actuelle

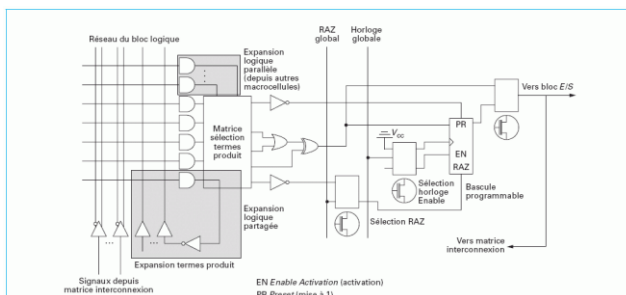
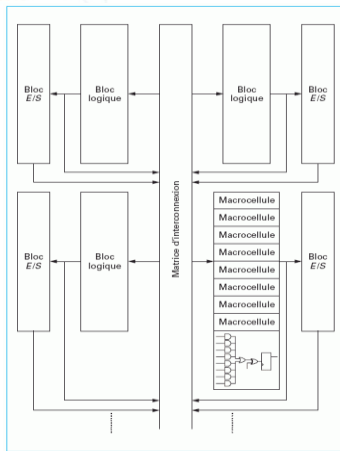


Figure 24 - Exemple de macrocellule d'un CPLD (série MAX3000 d'Altera)

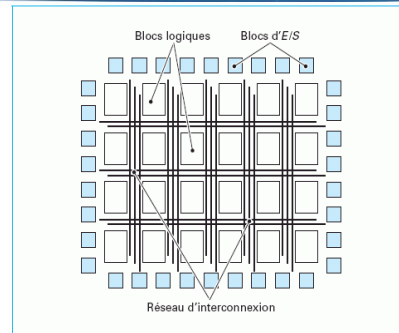
macrocellule de CPLD

40

40



structure CPLD



structure FPGA

FPGA plus interconnecté $\Rightarrow$ plus flexible

$\Rightarrow$ plus difficile à implémenter

$\Rightarrow$ délais plus difficiles à estimer

FPGA plus de ressources intégrées



Implémentation avec un logiciel et un langage de programmation

41

41

Fabricant CPLD : Intel/Altera, Lattice, Microchip



- Lattice LC4256ZE
- 20,41 € Mouser 2026
- 256 macrocellules

Fabricant FPGA : Amd/Xilinx, Intel/Altera, Lattice

- Intel /Altera cyclone 5 5CGXFC9
- 764,85 € Mouser 2026
- 301 000 blocs logiques,
- 14 Mbits RAM, DSP



- AMD/Xilinx Spartan 7 XC7S100
- 182,25 € Mouser 2026
- 102 400 blocs logiques,
- 5 Mbits RAM, DSP

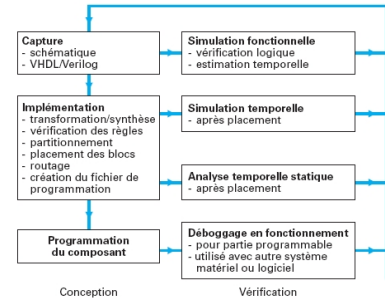
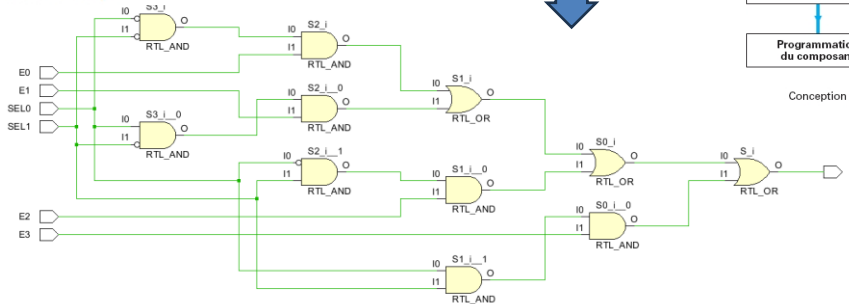


42

42

- VHDL, verilog

```
architecture Flow of MUX4 is
begin
    S <= ((not SEL0) and (not SEL1) and E0) or
          (SEL0 and (not SEL1) and E1) or
          ((not SEL0) and SEL1 and E2) or
          (SEL0 and SEL1 and E3);
end Flow;
```

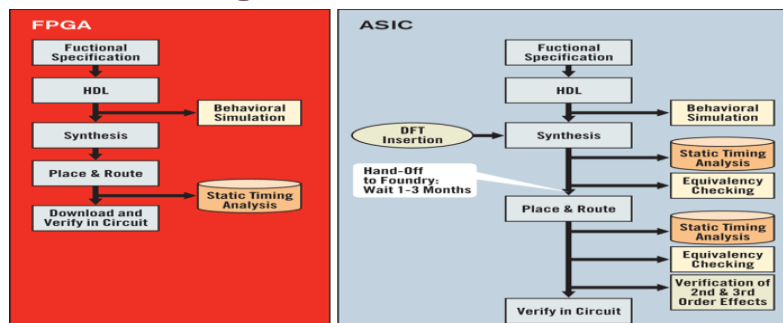
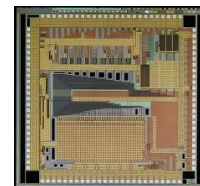


43

43

ASIC (Application Specific Integrated Circuit)

- Circuit dédié
- ⇒ plus performant 😊
 - ⇒ coût réduit pour des grandes quantités 😊
 - ⇒ temps de conception plus long 😞
 - ⇒ pas flexible 😞

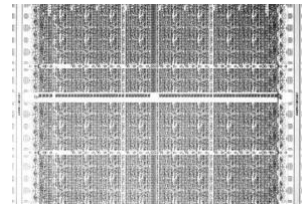


44

44

spécifique versus flexible

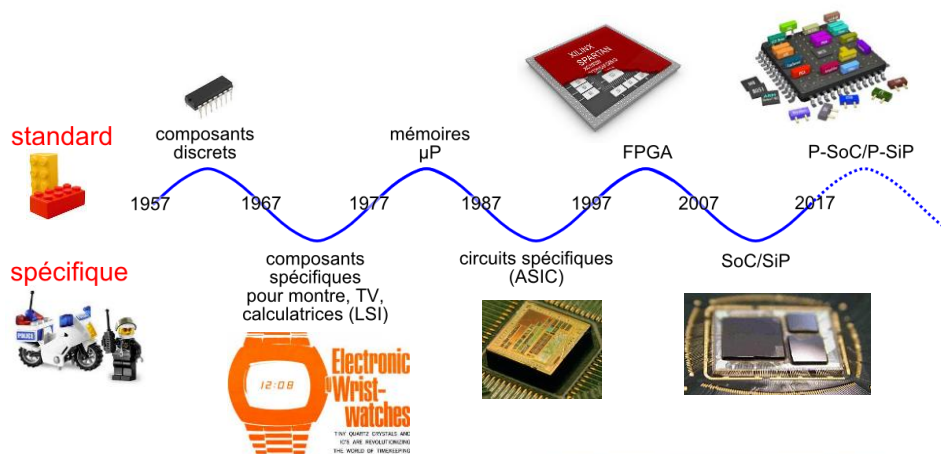
- | | |
|--|---|
| <ul style="list-style-type: none"> ❑ Permanent, ne peut plus être modifié ❑ Temps de conception long (plrs mois) ❑ <i>Non-recurring engineering (NRE) costs</i> (10^5-10^6 \$) ❑ Optimisé pour l'application (meilleures performances -vitesse, consommation-) ❑ Coût plus faible pour une production de grande ampleur | <ul style="list-style-type: none"> ❑ Peut être modifié (adaptation à un cahier des charges changeant) ❑ Temps de conception plus court (programmation rapide, test direct, moins de corps de métiers) ❑ Logiciels de développement peu onéreux |
|--|---|



45

45

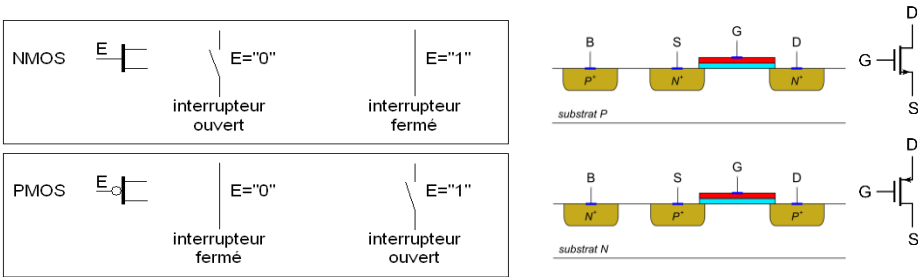
Spécifique versus flexible



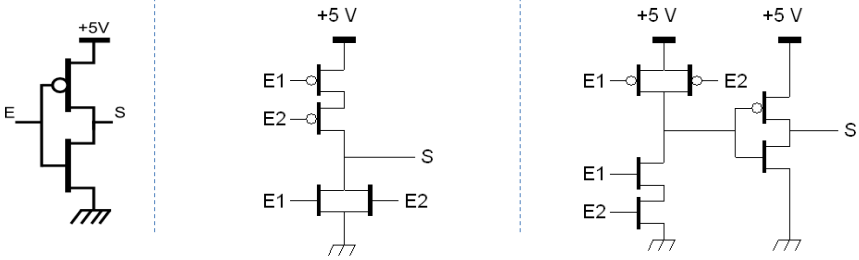
46

46

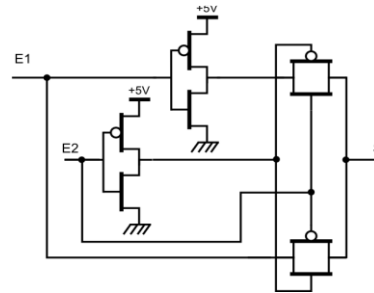
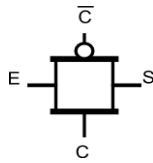
D'un point de vue physique, les portes logiques sont réalisées à partir de transistors. Plusieurs types de transistors peuvent être employés, mais le plus couramment utilisé est celui des transistors Métal Oxyde Isolant, plus connu sous le sigle **transistor MOS**. La fabrication des portes logiques s'appuie alors sur l'utilisation de deux types de transistors : les transistors **NMOS** et les transistors **PMOS**. Ces transistors sont utilisés comme des interrupteurs et chacun d'entre eux est caractérisé par les niveaux logiques qui le rendent assimilable à un **interrupteur ouvert ou fermé**.



Exemples de portes logiques



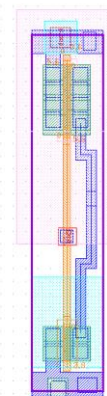
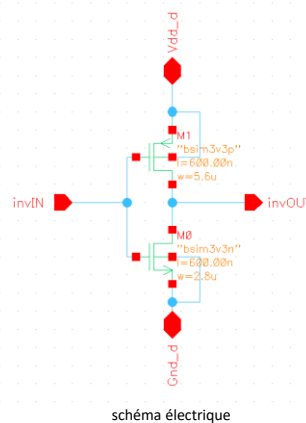
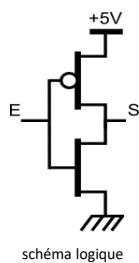
Exemples de portes logiques



49

49

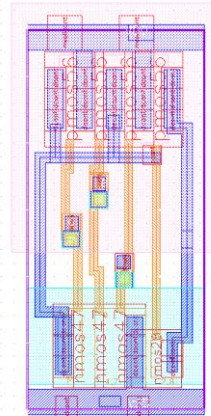
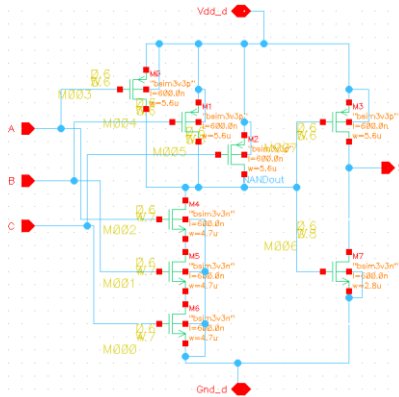
Exemples de layout de portes logiques



50

50

Exemples de layout de portes logiques (cont.)



51

51

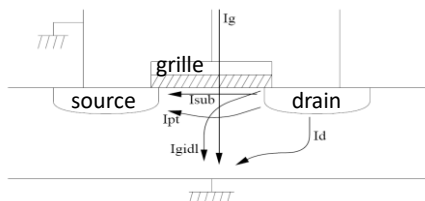
- Temps de commutation (delay)
- Puissance dissipée

— Deux aspects différents :

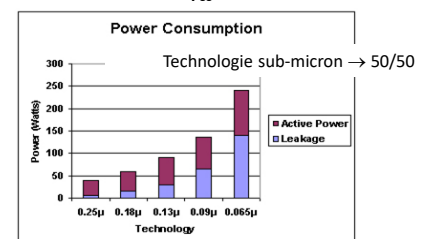
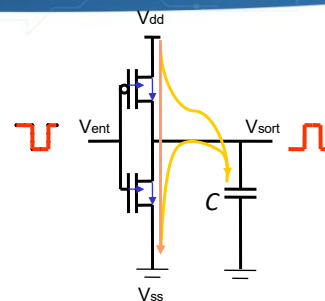
- Puissance dynamique : activité
- Puissance de fuite (leakage power)

$$\text{delay} \propto \frac{V_{dd}}{(V_{dd} - V_{th})^2}$$

$$P = C \cdot V_{dd}^2 \cdot F$$



Isub : courant de fuite sous le seuil (subthreshold off state leakage current)
Ig : courant de fuite par effet tunnel (gate tunnelling leakage current)
Id : courant de fuite de jonction inverse (reverse junction leakage current)
Igidl : courant de fuite induit par la grille (Gate Induced Drain Leakage)
Ipt : courant de fuite transistor à canal court (Drain Source Punch Through)



52

- JEDEC* committee
- JC-16 Interface Technology
- The activities within JC-16's scope include the specification of power supply voltage levels for digital integrated circuits and the definition of electrical interfaces between the components of a system. The committee scope further encompasses interface protocols, modeling, simulation, testing environments, and verification.

*Joint Electron Device Engineering Council

For each V_{CC} a standard is defined, with commonly agreed-upon levels of input and output levels. Figure 1 shows the appropriate switching levels for 5-V, 3.3-V, and 2.5-V that have passed the JEDEC committee. Additionally, the 1.8-V level specification, as given in the data sheets of the SN74VCMXXA and SN74AVCMXX devices, is shown.

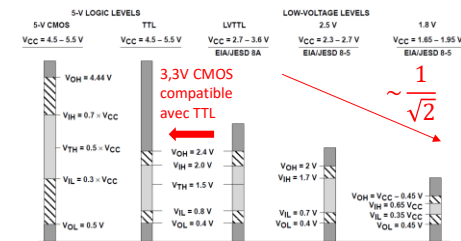


Figure 1. 5-V, 3.3-V, and 2.5-V Switching-Level Comparison

EIAJ ED-5001A

2.2 Recommended operating conditions

Table 2 Recommended operating conditions

Parameter	Symbol	Normal range (3.3V nominal)	Wide range (3V nominal)	Unit
Power supply voltage	V_{DD}	3.0 ~ 3.6	2.7 ~ 3.6	V
Operating temperature range	T_a	Note 3	Note 3	°C

Note 3: Specified by manufacturer for various purposes, respectively.

53

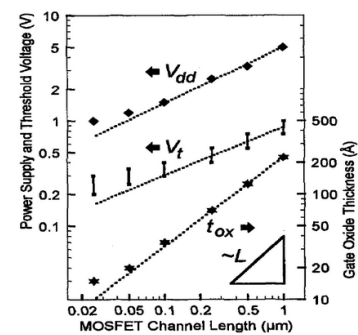
53

- Cela réduit la puissance active consommée

$$P = C \cdot V_{dd}^2 \cdot F$$

- Pour conserver un champ électrique constant lors de la miniaturisation.

$$E_{ox} = \frac{V_{dd}}{t_{ox}}$$



Évolution de la tension d'alimentation V_{dd} , de la tension de seuil V_{th} et de l'épaisseur d'oxyde en fonction de la technologie.

54

54



- Quel outil graphique permet de faciliter la synthèse de petits circuits logiques combinatoires ?
Quelle est sa limitation ?
- Comment fait-on la synthèse de circuit logique combinatoire sur les composants de type logique structurée ?
- Quels sont les avantages/inconvénients des solutions programmables de type FPGA et fixe ASIC ?