Faculté des Sciences Appliquées
Département Montefiore
SERVICE MICROÉLECTRONIQUE
Prof. J. Destiné.
A. Leroux, Assistante.
F. Senny, Assistant.

Le mardi 24 juin 2008

ELEN-040

Électronique numérique

Notes de laboratoire

Ulg – Département Montefiore, Sart-Tilman, Bât.28, Parking B32,
B-4000 Liège (Belgium). Fax : +32 4 366 29 50,
e-mails : destine@montefiore.ulg.ac.be ; degreve@montefiore.ulg.ac.be
tél : J. Destiné : +32 4 366 26 16 ; S. De Grève : +32 4 366 26 17
http://www.montefiore.ulg.ac.be/services/microelec/
http://www.montefiore.ulg.ac.be/~degreve/

Table des matières

Prologue

Ces notes de laboratoire constituent un complément indispensable au cours et aux répétitions de théorie. Il est primordial de prolonger l'enseignement par une expérimentation concrète qui illustre la théorie, justifie les modèles simplificateurs présentés et dissipe les appréhensions face au "hardware" en montrant combien est facile l'assemblage de circuits logiques, c'est-à-dire d'éléments constitutifs d'ordinateurs.

Le tableau brossé ici n'est pas volontairement idyllique. Il y a évidemment un monde de différences entre la réalisation d'un additionneur combinatoire et celle d'une carte programmable qui concentre l'ingéniosité de plusieurs années-homme. L'un comme l'autre cependant font appel aux mêmes composants de base. Le respect d'un cahier des charges par le mariage performant de logiciel et de matériel ne s'obtient que par une longue expérience qui débute au laboratoire didactique, ou mieux encore, si l'on est un mordu, dans son labo personnel.

Comme pour les mathématiques, et plus modestement, il n'y a pas de voie royale pour l'électronique! (Allusion à la réponse du brillant mathématicien Euler au tsar de Russie, dont il était l'hôte, et qui, habitué aux passe-droits que son titre impliquait, demandait une courte-échelle à la connaissance en mathématiques). Le chemin de la maîtrise commence par les manipulations élémentaires du laboratoire didactique.

Remarques :

Une bonne part de l'équipement du laboratoire a été acquise grâce à des subventions et aides du **Legs PISART**.

Que ce dernier en soit encore remercié ici.

D'autre part, que l'exemple de Monsieur PISART soit suivi par celles ou ceux d'entre vous qui réussiront professionnellement presque aussi bien (ou mieux) que lui, pour léguer, comme il l'a fait, 100 millions à la Faculté des Sciences Appliquées de son Alma Mater!

Bonne route !

Préface

Ces notes constituent une sixième version du manuel de laboratoire du cours d'électronique numérique (ELEN040) à l'intention des étudiants de première technique électricien et physicien ainsi que des 1ère licence en informatique.

Ces laboratoires et le travail qui l'accompagne sont **obligatoires** et demandent la remise d'un **rapport** et d'une **présentation** pratique.

Les expériences de laboratoire proposées illustrent l'ouvrage *"Computer Engineering Hardware Design"* de M. Mano vu au cours théorique et présentent les techniques largement répandues et plus modernes des composants programmables (PLD).

Ces laboratoires ont pour but de permettre aux étudiants d'améliorer leur perception "physique" du hardware des circuits et de se familiariser avec divers appareils de mesure utilisés en électronique: oscilloscope, analyseur logique…

Les annexes de ce manuel fournissent des résumés succincts de l'utilisation de ces appareils et rassemblent les informations permettant de se constituer une "boîte à outils" simple et peu onéreuse, mais fonctionnelle pour le design et le test de circuits digitaux.

Organisation pratique

1 Présentation des labos

Les laboratoires du cours d'électronique numérique sont constitués de 2 parties; chacune se déroulant par groupe de 3 étudiants.

Ces 2 parties sont :

- Une introduction aux composants de logique digitale
- Une introduction à la logique programmable (VHDL)

Les étudiants de première technique électricité effectueront donc une matinée de manipulation et un travail relatif à la logique programmable.

1.1 Introduction aux composants de logique digitale

Cette première partie se déroule au laboratoire R100 et est encadrée par un élève moniteur ou un assistant qui sera là pour vous aider et répondre à toutes vos questions.

Afin de réaliser efficacement toutes les manipulations demandées (**voir Chap V**), chaque groupe devra se procurer, avant la date du labo, une série de composants électroniques (voir liste ci-après). En ce qui concerne les appareils de mesure, ceux-ci sont disponibles sur les tables du labo. Une commande groupée est sans doute plus avantageuse et moins fastidieuse, quelqu'un de votre groupe devra se désigner pour s'en charger.

Vous pouvez trouvez ces composants notamment chez Micro-Select (Avenue Blonden, 42, 4000 Liège, tél. : +32 (0)4 252 42 32, www.micsel.be), ou encore chez fissette (www.fissette.com).

1.2 Introduction à la logique programmable

Cette deuxième partie consiste à programmer une GAL et à la tester électriquement (**voir chap IV et VI**).

Si vous possédez un PC, vous pouvez réaliser votre programme à domicile. Pour cela, vous devez installer le logiciel Warp (release 6.3) disponible sur CDROM dans le service de microélectronique. Si vous ne possédez pas de PC, vous pouvez utiliser ceux qui sont mis à votre disposition au R100. Il est également possible d'utiliser la suite ispLEVER de chez Lattice.

Remarque : Dans ce cas, veuillez garder vos fichiers sur disquettes et non sur le disque dur de l'ordinateur.

Lorsque vos simulations sur HDL-Sim sont correctes, vous pouvez alors passer à la programmation réelle du PLD. Pour cela, vous devez utiliser le programmeur du R100. Si vous éprouvez des difficultés, les manuels d'utilisation sont à votre disposition.

Une fois le chip programmé, il ne vous reste plus qu'à le tester électriquement. Ce test s'effectue au labo R100 avec le même matériel que celui utilisé pour les manipulations des composants digitaux.

Remarque : Le labo R100 est à partager avec d'autres sections. Vérifier sur l'horaire (affiché normalement sur la porte) qu'il est libre lorsque vous désirez faire vos essais.

2 Matériel

2.1 Composants à acheter

Pour réaliser les différents laboratoires, vous utiliserez des composants CMOS de la famille HC ou HCT. Les composants à se procurer sont les suivants :

- 74HC00 QUAD NAND 2-IN
- 74HC04 HEXA INV
- 74HC08 QUAD AND 2-IN
- 74HC10 TRI NAND 3-IN
- 74HC86 QUAD XOR 2-IN
- 74HC93 4 BIT COUNTER
- 74HC153 DUAL 4X1 MUX
- 74HC139 DUAL 1X4 DECODER
- NE555P TIMER 555
- 1 BOUTON POUSSOIR
- 10 LEDs
- 10 résistances de 330Ω
- 2 résistances de 10 kΩ
- 1 résistance de 100kΩ
- 2 condensateurs de 10nF

2.2 Disponible au labo

Chaque table du laboratoire dispose normalement de :

- Un multimètre digital
- Un oscilloscope
- Un générateur d'impulsions
- Une alimentation stabilisée

Pour vos montages, vous utiliserez un **breadboard** (planche à essais) que vous demanderez au laboratoire même. Demandez aussi des bouts de fils monobrins, des câbles coaxiaux, des pinces croco, …

Nous conseillons aux passionnés d'électronique l'achat d'un catalogue des différents composants. Les plus intéressants sont probablement ceux de Texas Instruments, Philips ou Motorola. (pour les moins férus, des catalogues sont également disponibles au laboratoire).

3 Rapport et présentation

Afin d'apprécier votre travail et d'évaluer votre compréhension du cours, nous vous demandons de :

- Remettre un rapport.
- Présenter oralement le circuit digital que vous avez réalisé.

3.1 Le rapport

Dans ce document, vous exprimerez vos connaissances sur les expériences réalisées, et ce, le plus clairement et le plus complètement possible. Complet ne veut pas dire parler pour ne rien dire. Ce n'est pas l'épaisseur qui compte, mais la finesse des observations, la validité de ce que vous y dites, l'intérêt des tests que vous avez pu essayer pour mieux comprendre et contrôler les phénomènes.

Montrez que vous avez compris, expliquez vos découvertes, vos surprises. Quand on vous demande de dessiner un circuit, schématisez-le, développez les équations…

Ce rapport comportera également une partie concernant la logique programmable. Vous décrirez le problème que vous avez choisi, la manière dont vous avez traité, les difficultés rencontrées... Vous y adjoindrez également votre programme (commenté) ainsi que vos simulations.

Vous pouvez aussi donner par un e-mail à votre assistant des suggestions sur le déroulement de ces labos, leur intérêt, d'éventuelles améliorations à apporter aux notes…

3.2 La présentation

Lors de celle-ci, vous expliquerez aux assistants le problème que vous avez résolu et vous leur simulerez, à l'aide des appareils mis à votre disposition, quelques cas de figure.

L'appréciation de la présentation dépendra principalement de l'originalité du travail, de la manière dont les simulations sont menées, de la clarté de vos explications …

Composants passifs

1 Les Résistances

La valeur d'une résistance peut se lire en décodant les anneaux de couleur tracés sur celle-ci. Toutes les valeurs n'existent pas (voir tableau fourni en annexe).

Pour connaître la valeur d'une résistance, diriger la bague argentée (tolérance 10%) ou dorée (tolérance 5%) vers la droite (le premier anneau est toujours le plus proche du bord), parfois, il faut placer la bande la plus épaisse à droite. Lire ensuite trois (ou quatre) chiffres suivant la table :

0	noir	5	vert
1	marron	6	bleu
2	rouge	7	violet
3	orange	8	gris
4	jaune	9	blanc

Tableau 1 : code des couleurs

Pour un code à trois bagues, le premier chiffre est celui des dizaines, le second, celui des unités et le dernier, l'exposant de 10

Exemple : orange, orange, marron, argent : 33.101 = 330Ω ± 10%

Il faut aussi veiller à la puissance dissipée dans la résistance. C'est la taille physique de celle-ci qui indique sa puissance maximale. Les plus petites acceptent jusqu'à 0,25W.

Une méthode souvent plus rapide pour mesurer la valeur consiste à mesurer celle-ci à l'aide d'un multimètre (remarque : ne pas mesurer la valeur d'une résistance dans un circuit alimenté).

Un tableau plus complet du code couleur est affiché au labo et un autre est fourni en annexe 7.

2 Les Diodes

Différents types de diodes existent sur le marché. Citons notamment la diode à jonction, la diode Zéner, la diode électroluminescente (LED), …

Les diodes utilisées lors de ce labo sont des LED. Leur fonctionnement est le suivant :

- polarisée **en direct** (sens du courant = sens de la flèche), elle s'illumine et la tension à ses bornes est de 1,6V. Le courant la traversant doit être limité à 20mA.
- polarisée **en inverse**, la LED reste éteinte et elle se comporte comme un circuit ouvert.

Pour visualiser un état logique dans un circuit CMOS, on la met généralement **en série** avec une résistance de 330Ω. Le sens de branchement se reconnaît à la longueur des pattes et au méplat du bulbe, petite patte du coté de plus faible tension :

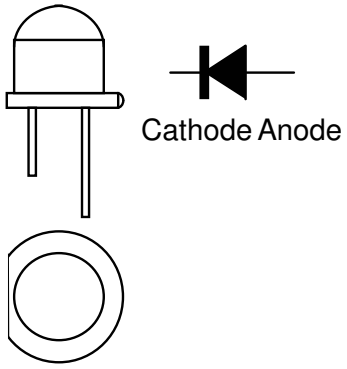


Figure 1

3 Les Condensateurs

Les valeurs des capacités s'échelonnent de quelques picofarads (10^{-12} F) à quelques milliers de microfarads (10^{-6} F). Il existe deux grandes familles de condensateurs :

- les condensateurs non polarisés (10^{-12} F-> 10^{-6} F)
- les condensateurs électrolytiques (cylindres bleus) dont les valeurs peuvent atteindre plusieurs milliers de microfarads.

Les condensateurs utilisés au laboratoire sont généralement non polarisés. Le sens du branchement n'a donc pas d'importance. Par contre, pour les condensateurs électrolytiques, les signes sont clairement indiqués près des pattes. Une erreur de branchement provoque souvent une explosion du composant avec les conséquences que cela suppose : brûlures, détérioration d'autres composants…

Une autre caractéristique importante d'un condensateur est sa tension maximale admissible. Celle-ci est généralement inscrite sur le cylindre

Remarque : la valeur de la capacité peut être codée en couleurs ou être directement inscrite en µF (le plus courant actuellement).

Les familles de circuits intégrés

1 Présentation des IC

1.1 Introduction

Selon leur taille, les circuits intégrés peuvent être classifiés comme **SSI** (Small Scale Integration), **MSI** (Medium Scale Integration), **LSI** (Large Scale Integration) ou **VLSI** (Very LSC). Les SSI contiennent moins de 12 portes logiques (ou portes équivalentes); les MSI peuvent contenir jusqu'à 100 portes équivalentes et les LSI et VLSI intègrent plus de 10000 portes, ce qui permet de créer des fonctions logiques plus complexes.

Les circuits intégrés utilisés lors de ce laboratoire seront de type SSI. Ils possèderont entre 14 et 16 broches d'entrée/sortie. La numérotation des pattes de tous les ICs respecte la même convention : en dirigeant les pattes vers le bas et l'encoche vers le haut, la patte en haut à gauche est numérotée 1. Descendez ensuite en parcourant le périmètre du boîtier et vous trouvez les broches de numéro croissant.

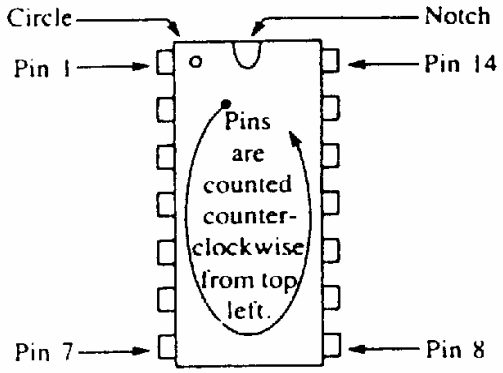


Figure 2 : Numérotation des broches

Un IC est différencié d'un autre par un numéro d'identification. Par exemple, le circuit **74LS00** est une quadruple porte NAND à 2 entrées :

- '74' : standard le plus utilisé actuellement
- 'LS' : technologie de fabrication (Low power Schottky)
- '00' : type de circuit (00 = quad 2-input NAND).

1.2 Alimentation

Chaque IC doit être alimenté. Pour cela, connectez la broche VCC à une tension de 5V stable et la broche GND à la masse correspondante. En général, la masse se situe en bas, à gauche (pin 7 ou 8) et la tension positive, à l'opposé du chip (pin 14 ou 16). Cette règle n'est pas toujours respectée, il est donc prudent de vérifier avant de raccorder. Une **inversion des bornes d'alimentation** est souvent **fatale**.

Si vous disposez d'une alimentation stabilisée de 5V (comme celle du labo), utilisez-la. A défaut, voyez l'annexe qui vous montrera comment en construire une très simplement.

1.3 Entrées

Une entrée logique est une tension. La valeur '0' est implémentée par une connexion à la masse. Un '1' correspond normalement à une connexion au VCC. Il est toutefois conseillé dans les circuits "sérieux" d'introduire une résistance de 1kΩ **en série** afin de protéger l'entrée contre les surtensions transitoires. Cette résistance suffit pour 50 entrées. Il ne faut jamais laisser d'entrées flottantes, surtout avec des circuits CMOS, puisque une entrée flottante est une entrée indéterminée.

Si vous voulez alternativement entrer un '1' et un '0', vous pouvez utiliser un interrupteur monté comme suit :

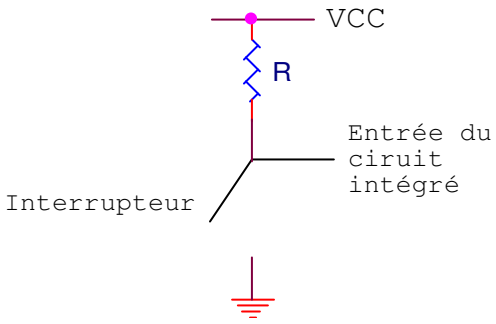


Figure 3 : raccordement d'un interrupteur

Si l'interrupteur est en position ouverte (comme sur la figure), le circuit "voit" en entrée un '1' logique. S'il est en position fermée (raccordé à la masse), le circuit "voit" un '0' logique. La résistance, dite résistance tire-haut, a généralement une valeur comprise entre 5 et 10 kΩ.

1.4 Sorties

Un premier moyen de vérifier une valeur logique est l'utilisation des sondes d'un **multimètre**. La sonde noire peut rester connectée en permanence à une masse, la sonde rouge sera mise en contact avec le fil portant le niveau logique à lire.

Pour vérifier plusieurs valeurs logiques simultanément ou pour mesurer des valeurs transitoires, cette méthode devient lourde et il devient plus simple d'utiliser une méthode de mesure plus visuelle, comme par exemple, les LEDs.

Comme rappelé au chapitre 2, une LED ne conduit et ne s'illumine que si elle est polarisée en sens direct. La tension à ses bornes est alors voisine de 1,6V. Une LED ne peut être reliée directement à une tension de 5V. On la placera donc toujours en série avec une résistance de ± 330Ω. Cette résistance verra à ses bornes une tension de 5V - 1,6V = 3,4V, elle sera donc parcourue par un courant de : 3,4V/330 Ω =10mA, comme la diode. Ce courant est suffisant pour allumer la LED, mais pas trop important pour ne pas l'endommager. Si la diode est polarisée en inverse, elle se comporte comme un circuit ouvert jusqu'à une certaine valeur de la tension inverse (avalanche).

Si vous préférez voir la diode allumée pour l'autre valeur logique, insérez un inverseur à la sortie du circuit.

Une troisième façon de mesurer l'état d'une sortie est d'utiliser l'**analyseur logique**. Celui-ci enregistre des signaux logiques en parallèle pendant un temps qui est fonction de sa capacité mémoire et de la fréquence d'échantillonnage. Ces signaux ne doivent pas obligatoirement être périodiques, comme sur un oscilloscope. Le principe de fonctionnement de l'analyseur est fourni en annexe.

2 Terminologie

Malgré le grand nombre de fabricants de circuits intégrés, la terminologie utilisée est presque normalisée. Afin de comprendre un peu mieux la description des circuits qui est donnée dans les datasheets (voir annexe), voici l'explication des termes les plus usités.

- **V_{IH}** (Tension d'entrée niveau HAUT) : le niveau de tension nécessaire pour avoir un 1 logique en entrée. Toute tension inférieure à ce niveau n'est pas considérée comme état HAUT par le circuit logique.
- **V_{IL}** (Tension d'entrée niveau BAS) : le niveau de tension nécessaire pour avoir un 0 logique en entrée. Toute tension supérieure à ce niveau n'est pas considérée comme état BAS par le circuit logique.
- **V_{OH}** (Tension de sortie niveau HAUT) : le niveau minimum de tension de la sortie d'un circuit logique correspondant à l'état logique 1.
- **V_{OL}** (Tension de sortie niveau BAS) : le niveau maximum de tension à la sortie d'un circuit logique correspondant à l'état logique 0.

Les champs électriques et magnétiques parasites peuvent induire des tensions dans les fils de raccordement des circuits logiques. Ces signaux sont appelés **bruits** et peuvent parfois amener la tension sous la valeur V_{IH} ou la porter au dessus de V_{IL}. L'immunité au bruit d'un circuit logique définit l'aptitude du circuit à tolérer des tensions parasites sur ses entrées. La mesure quantitative de l'immunité aux bruits est appelée **la marge de sensibilité aux bruits**. Elle est illustrée sur la figure suivante par les symboles **V_{NH}** et **V_{NL}**.

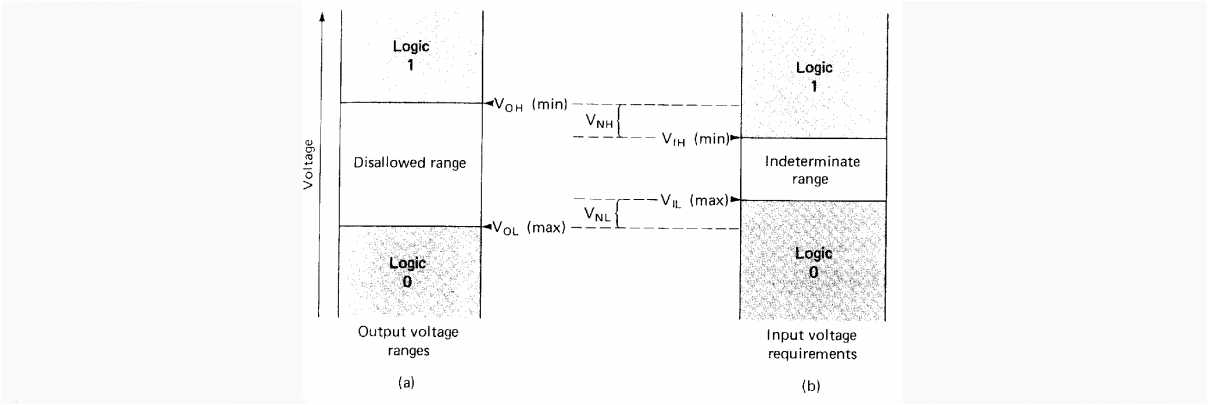


Figure 4 : Marges de sensibilité aux bruits

Toutes les tensions qui se situent dans la bande indéterminée ne doivent jamais apparaître sur les bornes d'un circuit logique car elles occasionnent une réponse imprévisible.

D'après le schéma ci-dessus, la **marge de sensibilité aux bruits état haut** V_{NH} est définie par :

$$V_{NH} = V_{OH} - V_{IH}$$

Et la **marge de sensibilité aux bruits état bas** V_{NL} :

$$V_{NL} = V_{IL} - V_{OL}$$

Les retards de propagation : un signal qui traverse un circuit subit toujours un retard. Deux retards de propagation sont définis :

- **t_{PLH}** : retard pour passer du niveau logique 0 au niveau logique 1

- **t_{PHL}** : retard pour passer du niveau logique 1 au niveau logique 0

Ces retards sont illustrés sur la figure suivante :

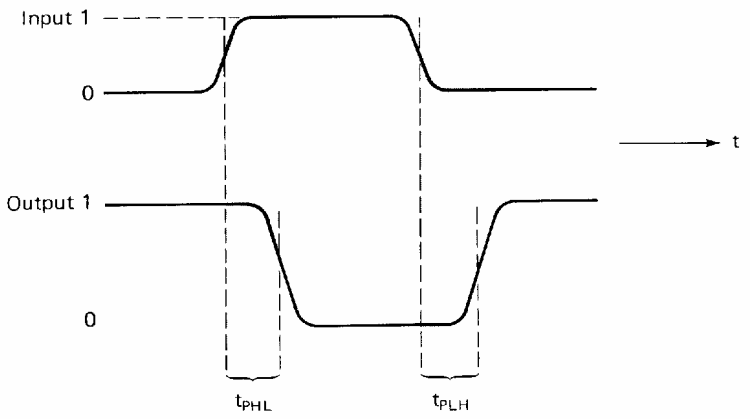


Figure 5 : Les retards de propagation

La **sortance** *(fanout)* est le nombre maximum d'entrées logiques que la sortie d'un même dispositif est capable de piloter sans problème. Si on dépasse le nombre indiqué, il n'est pas assuré que les tensions des niveaux logiques des sorties seront exactes. Il est donc très important de connaître le fanout de tous les circuits utilisés dans un montage. Les portes ayant le fanout le plus élevé sont les buffers.

3 Les familles TTL

Les familles TTL sont commercialement disponibles depuis la fin des années 1960. Ses avantages sont un coût très bas, une vitesse élevée, une possibilité d'interconnexion *(drive-capability)*, une immunité aux bruits acceptable ainsi que la disponibilité de centaines de composants compatibles. Ceci donne une large sélection de blocs logiques simples ou élaborés qui peuvent être directement interconnectés pour produire des fonctions logiques plus compliquées.

Les composants TTL peuvent se brancher directement l'un à l'autre, sans se préoccuper des caractéristiques électriques. Ce niveau d'abstraction peut être atteint si l'on respecte certaines règles simples de raccordement qui ont été fixées à la conception par des études des niveaux de tension et des échanges de courant au niveau des transistors d'entrée ou de sortie.

De toutes les familles TTL, les séries **54/74** sont les plus utilisées**.** La différence entre ces 2 familles tient aux plages de températures et de tensions (plus large pour la série 54). De nombreux fabricants de semiconducteurs produisent des circuits TTL. Heureusement, ils acceptent tous la même numérotation et ainsi, le même numéro de circuit intégré correspond toujours au même élément. Chaque fabricant ajoute juste un préfixe. Par exemple, Texas Instrument utilise le préfixe SN, Motorola MC ...

La tension d'alimentation nominale pour du TTL est de 5 volts. En ce qui concerne la plage de température, la série 74 fonctionne correctement entre 0 et 79°C et la série 54 admet des températures allant de -55 à 125°C.

Une **entrée non-connectée** en TTL se comporte comme si la valeur qui lui était appliquée était un **1 logique** puisque la jonction émetteur-base n'est pas polarisée.

Après l'avènement de la série 74 standard, d'autres séries TTL ont été mises au point. Parmi celles-ci, citons :

- Série **74L** : famille TTL à faible consommation.
- Série **74H** : famille TTL rapide.
- Série **74S** : famille TTL construite avec des transistors Schottky.
- Série **74LS** : famille TTL Schottky faible consommation.
- Série **74AS** : famille TTL Schottky avancée.
- Série **74ALS** : famille TTL Schottky avancée faible consommation.

Le tableau suivant permet de comparer les paramètres qui caractérisent chacune des familles TTL.

	74	**74L**	**74H**	**74S**	**74LS**	**74AS**	**74ALS**
Valeurs des Performances							
Retard de propagation (ns)	9	33	6	3	9.5	1.7	4
Consommation (mW)	10	1	23	20	2	8	1.2
Fréquence d'horloge max (MHz)	35	3	50	125	45	200	70
Fanout (même série)	10	20	10	20	20	40	20
Paramètres de tension							
VOH (min)	2.4	2.4	2.4	2.7	2.7	2.5	2.5
VOL (max))	0.4	0.4	0.4	0.5	0.5	0.5	0.4
VIH (min)	2	2	2	2	2	2	2
VIL (max)	0.8	0.7	0.8	0.8	0.8	0.8	0.8

Tableau 2 : Caractéristiques des séries TTL

4 Les familles CMOS

A l'heure actuelle, la famille CMOS (Complementary Metal Oxyde Semiconductor) s'impose de plus en plus dans les dispositifs MSI, principalement aux dépens de la famille TTL. Le procédé de fabrication des CMOS est plus simple que celui des TTL et sa densité d'intégration est plus élevée. Les CMOS ne consomment qu'une fraction de l'énergie dissipée par la série TTL faible consommation (74L). Normalement, la vitesse de fonctionnement des CMOS est inférieure au TTL, mais la nouvelle série CMOS rapide concurrence désormais les séries 74 et 74LS.

Le circuit le plus simple en CMOS est l'inverseur. Il est constitué de deux transistors MOS en série; le dispositif canal P a sa source connectée à V_{dd} et le dispositif à canal N a sa source connectée à V_{ss}. Les grilles des deux dispositifs sont réunies pour former une entrée commune. Les drains sont également réunis pour former une sortie commune.

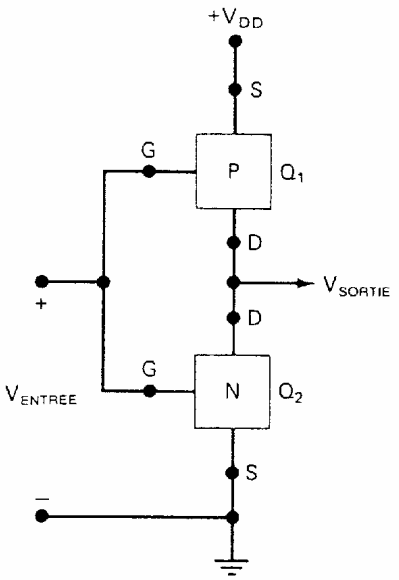


Figure 6 : L'inverseur CMOS

On retrouve un grand nombre de séries différentes dans les CMOS. La première série à voir le jour fut la série **4000**. La série **4000B** est une variante améliorée fournissant un plus fort courant de sortie. A ces familles de base, viennent s'ajouter de nouvelles séries :

- Série **74C** : compatible broche à broche avec la série TTL de même numéro et réalise les mêmes fonctions. Les performances de cette série sont à peu près les mêmes que celle de la série 4000.
- Série **74HC** : c'est une amélioration de la série 74C. La vitesse de commutation a été améliorée d'un facteur dix. La vitesse de ces dispositifs est comparable à celle de la série 74LS.
- Série **74HCT** : série CMOS rapide. La différence avec la série précédente est que celle-ci a été étudiée pour être compatible en tension avec les séries TTL, c-à-d que l'on peut les raccorder directement à des dispositifs TTL.

Les séries 4000 et 74C fonctionnent pour des valeurs de V_{dd} comprises entre 3 et 15V. Les séries 74HC et 74HCT fonctionnent pour des tensions allant de 2 à 6 volts. Ainsi, lorsque dans une même application, on désire utiliser des circuits TTL et CMOS, on utilisera une tension d'alimentation de 5 volts (puisque celle-ci est compatible CMOS et TTL).

En CMOS, les valeurs de V_{OL} et V_{OH} sont proches de 0 volt et de 5 volts. Pour ce qui est de V_{IL} et V_{IH}, on exprime leur valeur sous forme de pourcentage : V_{IL} = 30% de V_{dd} et V_{IH} = 70% de V_{dd}.

Les **entrées** CMOS ne doivent **jamais rester non branchées**. Toutes les entrées doivent être raccordées à une tension fixe ou à une autre entrée.

Le tableau suivant compare les caractéristiques des principales séries de circuits intégrés TTL et CMOS.

	74HC	4000B	74	74S	74LS	74AS	74ALS
Consommation par porte (mW)							
statique							
à 100 kHz	0.0025	0.001	10	20	2	8	1.2
	0.17	0.1	10	20	2	8	1.2
Retard de propagation (ns)	8	50	9	3	9.5	1.7	4
Vitesse - Consommation (mW)	10	1	23	20	2	8	1.2
Fréquence d'horloge max (MHz)	40	12	35	125	45	200	70
Marge aux bruits (cas pessimiste)	0.9	1.5	0.4	0.3	0.3	0.3	0.4

Tableau 3 : Comparaison des circuits intégrés TTL et CMOS

5 Interfaçage entre IC

On appelle **interfaçage** la connexion des sorties d'un circuit aux entrées d'un autre circuit dont les caractéristiques électriques sont différentes. Si les caractéristiques électriques des 2 circuits sont trop différentes, on doit avoir recours à un circuit d'**interface** que l'on placera entre les deux dispositifs. Le rôle de ce circuit sera de recevoir le signal de sortie du premier circuit et de le transposer afin qu'il devienne compatible avec les signaux du deuxième circuit.

Les CI d'une même famille sont conçus pour être raccordés ensemble (à condition que la sortance pour chaque sortie ne soit pas dépassée). Par contre, lorsque vous connectez la sortie d'un CI à l'entrée d'un CI d'une autre famille, vous devez prendre en considération les tensions des 2 éléments et donc consulter les datasheets. Le tableau suivant résume les valeurs des niveaux de tensions et de courants pour les familles les plus utilisées.

Paramètres	MOS			TL			
	4000B	**74HC**	**74HCT**	**74**	**74LS**	**74AS**	**74ALS**
VOH (min)	3.5V	3.5V	2V	2V	2V	2V	2V
VOL (max))	1.5V	1V	0.8V	0.8V	0.8V	0.8V	0.8V
VIH (min)	4.95V	4.9V	4.9V	2.4V	2.7V	2.7V	2.7V
VIL (max)	0.05V	0.1V	0.1V	0.4V	0.5V	0.5V	0.4V
IIH (max)	1µA	1µA	1µA	40µA	20µA	200µA	20µA
IIL (max))	1µA	1µA	1µA	1.6mA	0.4mA	2mA	100µA
IOH (max)	0.4mA	4mA	4mA	0.4mA	0.4mA	2mA	400µA
IOL (max)	0.4mA	4mA	4mA	16mA	8mA	20mA	8mA

Tableau 4 : Niveaux de tension et de courant des familles TTL et CMOS

5.1 CMOS piloté par TTL

Si l'on observe le tableau précédent, on remarque que les tensions de sortie hautes TTL ne sont pas compatibles avec les tensions d'entrées hautes CMOS (pour les niveaux bas, les 2 familles sont compatibles). On voit en effet que V_{OH} de toutes les familles TTL (2V) est plus bas que le paramètre V_{IH} des séries 4000B et 74HC (4,9V). Pour de telles associations, il est donc nécessaire de relever le niveau de tension de la sortie TTL pour l'amener à un niveau acceptable par l'entrée CMOS.

Une solution d'interfaçage est illustrée à la figure suivante. Grâce à une résistance Rp (1 à 10 kΩ) reliée aux 5 volts, on augmente la tension haute de sortie jusqu'à une valeur proche de 5 volts et donc compatible avec V_{IH} du CMOS.

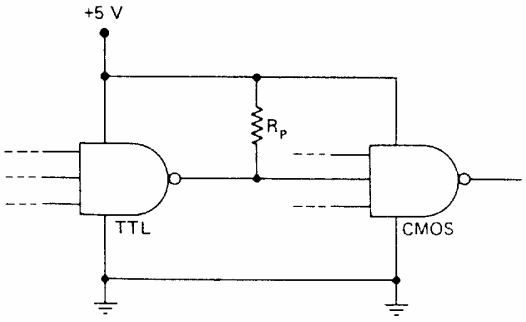


Figure 7 : Interfaçage entre circuits TTL et CMOS

Si l'on désire ne pas utiliser de circuits d'interfaçage, il faut utiliser la famille HCT qui est, quand à elle, compatible TTL sans précaution particulière.

5.2 TTL piloté par CMOS

TTL à l'état haut : d'après le tableau précédent, on peut voir que les sorties CMOS fournissent suffisamment de tension (V_{OH}) pour répondre au besoin d'un circuit TTL (V_{IH}). Aucun interfaçage ne doit donc être envisagé.

TTL à l'état bas : d'après ce même tableau, on constate que les courants d'entrée à l'état bas du TTL sont relativement intenses (100µA - 2mA). Les familles 74HC et 74HCT peuvent consommer jusqu'à 4mA et ne manifestent donc aucune difficulté à piloter un *seul* TTL de n'importe quelle série. Par contre, la série 4000B dont I_{OL} (0,4mA) est beaucoup plus faible et ne peut, en aucun cas, piloter une entrée d'un circuit appartenant à la série 74 ou 74AS.

Les circuits programmables

1 Introduction

Lors de la mise en pratique de diagrammes logiques et malgré la simplicité du principe, le concepteur s'expose à un risque d'erreurs important dû au nombre de fils à mettre en place et à vérifier. On se rend également compte de la vulnérabilité de ces réalisations aux faux contacts des breadboards et de la difficulté de corriger une éventuelle erreur.

Les composants à logiques programmables (**Programmable Logic Devices, FPGA, CPLD**) remédient à ces inconvénients d'une manière assez élégante. Ils permettent sur un seul circuit d'implémenter des fonctions aussi bien séquentielles que combinatoires. Le nombre de portes contenues dans ces composants allant de 150 pour les PLD à plus de 50000 pour les FPGA.

Ces circuits sont programmés à partir d'un fichier d'un format standardisé JEDEC. Ces fichiers sont difficilement interprétables par un opérateur humain car ils ne contiennent pratiquement que des 1 et des 0 spécifiant l'état désiré des fusibles. Le programmateur disponible au laboratoire permet de programmer des PLD et des CPLD de petites tailles. En ce qui concerne les FPGA, celles-ci peuvent être réalisées via des logiciels spécifiques au fabricant (Xilinx, Altera, …).

Des langages de conception de circuits programmables sont apparus pour aider l'utilisateur à créer des fichiers JEDEC à partir de description de schémas logiques dans un format ressemblant à un langage de haut niveau. Parmi ces langages, VHDL et Verilog sont les plus répandus. Le software (WARP dans le cadre de ce labo, ou ispLEVER Classic) va donc compiler le texte source VHDL de l'utilisateur et permettre de simuler le circuit décrit…

2 Les types de composants programmables

2.1 Les PLDs

Ce sont les composants programmables les plus populaires en industrie. Ils sont tous constitués d'un tableau de portes AND et d'un tableau de portes OR. Parmi les PLDs, on trouve notamment les **PALs** (programmable array logic) ou plus actuellement **GAL**.

Dans une PAL (la GAL est quasi équivalente), le tableau de AND peut être programmé alors que le tableau de OR est fixé. La figure suivante illustre la structure d'une sortie d'une PAL séquentielle. On remarque que la sortie peut contenir au maximum 8 produits, chaque produit dépendant des fusibles qui sont brûlés.

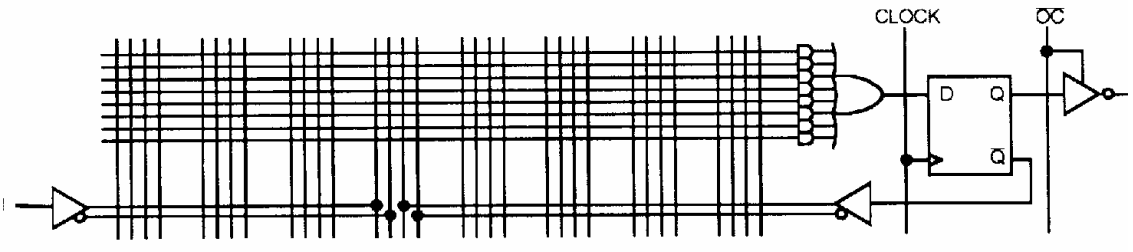


Figure 8 : Structure d'une PAL séquentielle.

De nombreuses PLDs existent sur le marché, elles se différencient par les paramètres suivants :

- le nombre maximum de pattes d'entrée,
- nombre de pattes de sortie ou d'entrée/sortie configurables,
- polarité des sorties (haute, basse, programmable),
- présence de registres de type fixe ou configurable, synchrones ou non, avec preset/reset synchrone ou non,
- complexité de la logique interne (registres cachés, feedbacks, facilité de test),
- temps d'accès,
- consommation en fonctionnement (les entrées changent) ou au repos,
- sorties à haute impédance commandables au niveau de chaque sortie ou de façon globale,
- possibilité d'effacer et de reprogrammer (U.V. ou électriquement).

Chaque PLD possède un numéro que l'on peut interpréter comme suit.

Exemple : AmPALCE22V10H-5PC

- 'Am' : sigle du fabriquant : TI (Texas Instrument), CY (Cypress) ,…
- 'PAL' : Programmable Array Logic : type de composant programmable,
- 'CE' : CMOS effaçable électriquement. Indique une technologie,
- '16' : nombre maximum de facteurs dans un produit (= nombre de pattes d'entrée dans un circuit sans feedback, nombre d'I/O si feedback),
- 'R' : type de sortie; R = registres, D = synchrones, A = registres à horloge programmable, X = XOR de deux sommes de produits, V = output logic macrocell,
- '8' : nombre d'unités de sortie.

Vous trouverez en annexe la signification des derniers symboles ainsi que les autres combinaisons existantes.

2.2 Les CPLDS

Les "complex" PLDs étendent le concept des PLDs à un plus haut niveau d'intégration de manière à améliorer les performances du système. Au lieu de concevoir des PLDs plus larges avec plus d'entrées, plus de termes de produits et plus de registres, une CPLD contient plusieurs blocs logiques, chacun semblable à une petite PLD. Les blocs communiquent les uns avec les autres via une matrice d'interconnexion (voir figure suivante).

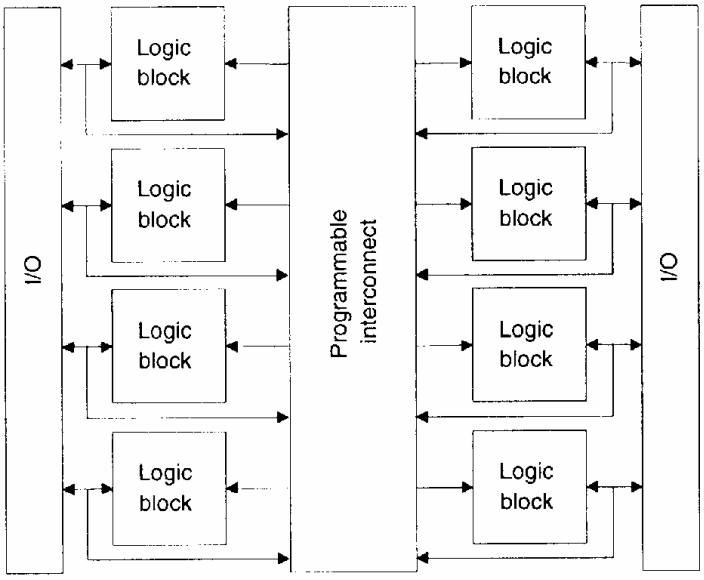


Figure 9 : Architecture d'une CPLD

Chaque bloc logique est alors constitué comme une GAL. On retrouve le réseau des produits, des sommes et des macrocellules.

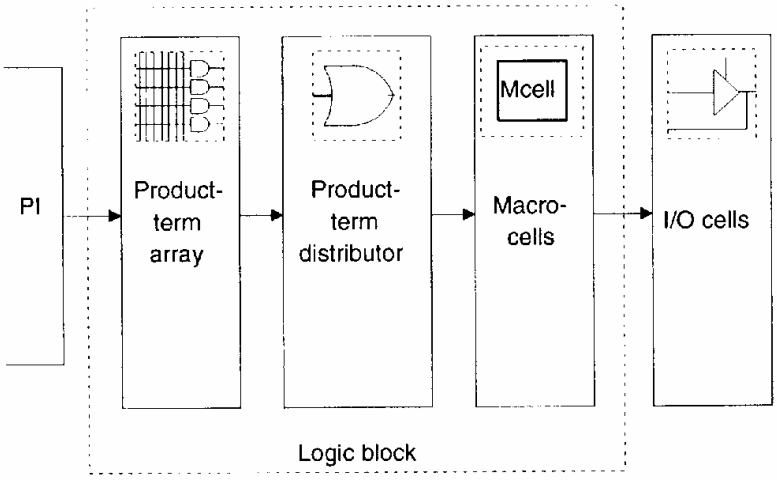


Figure 10 : Structure interne d'un bloc logique

La complexité, le nombre de macrocellules dans un bloc ainsi que le nombre de blocs varient d'un composant à l'autre. Chaque fabricant a également sa propre approche pour réaliser la distribution des produits. Ainsi, selon le circuit à réaliser, il est toujours intéressant de jeter un œil sur les datasheets pour trouver le composant le plus approprié pour l'application envisagée.

Parmi les grands fabricants de CPLDs, citons CYPRESS (avec la famille des FLASH370), ALTERA (avec la famille des MAX), Amd (avec la famille des MACH),…

2.3 Les FPGAs

L'architecture interne des FPGAs (Field Programmable Gate Array) est un tableau de petites cellules logiques qui communiquent les unes avec les autres via un réseau de « canaux de routage ».

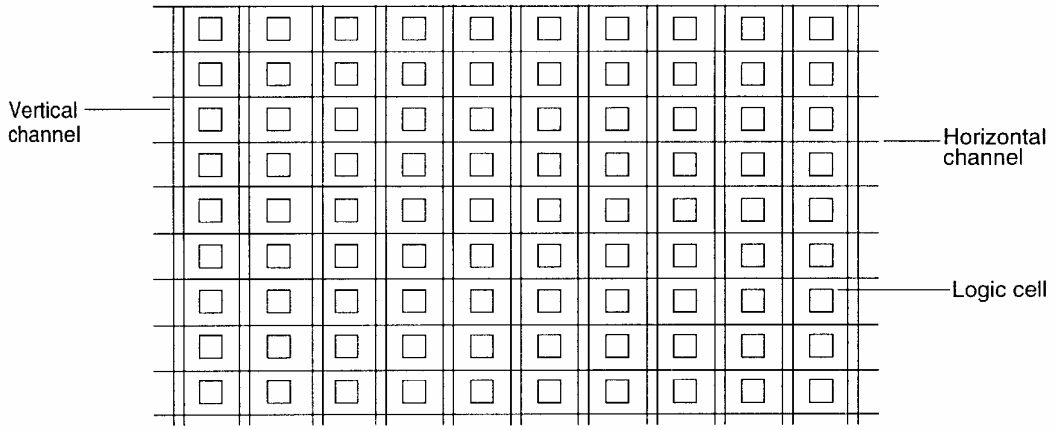


Figure 11 : Architecture d'un FPGA

Comparés aux CPLDs, les modules logiques des FPGAs sont plus réduits mais beaucoup plus nombreux. Pour réaliser une fonction complexe, on cascade alors plusieurs blocs logiques entre eux. Les interconnexions entre modules ne sont pas centralisées comme dans les CPLDs. Ils peuvent être vus comme une multitude de segments métalliques continus pouvant être reliés entre eux ou connectés en entrée et en sortie des blocs logiques.

Il existe deux technologies de FPGA : les SRAMs (Altera, Xilinx, atmel, …) et les antifuses.(Xilinx, Cypress, Actel, …). Le choix de l'une ou l'autre techno dépendra de nouveau de l'application envisagée.

3 Présentation de la GAL22V10

Dans ce laboratoire, nous nous limiterons à programmer la GAL22V10. Cette PLD est une des plus complètes et des plus répandues en ce sens qu'elle est configurable à différents niveaux et peut émuler (imiter le fonctionnement) de nombreuses autres.

En observant le schéma interne (voir figure suivante), on retrouve un plan de fusibles AND/OR (somme de produits) classique suivi de 10 OLMs (Output Logic Macrocells) et de buffers trois-états commandés par des produits et des broches d'entrée/sortie.

Toute la flexibilité provient de la conception de ces OLMs : en programmant seulement deux fusibles par OLM, la macrocellule se comporte comme une sortie combinatoire à polarité programmable ou comme un flip-flop D.

C1	C0	Description
0	0	Registre/actif bas
0	1	Registre/actif haut
1	0	Combinatoire/actif bas
1	1	Combinatoire/actif haut

Tableau 5 : Configuration des macrocellules

Les sorties tristates permettent de configurer chaque broche I/O comme entrée fixe, sortie fixe, sortie trois-états, ou entrée/sortie en fonction d'une condition interne.

Une autre originalité de la GAL22V10 est la distribution variable de produits (de 8 à 16). Une meilleure utilisation (remplissage) est ainsi normalement obtenue. Il peut ainsi arriver qu'un design refusé une première fois, soit accepté sous la seule condition d'échanger quelques pins.

L'horloge commandée par le flanc est commune à tous les registres. Ceux-ci disposent en plus d'un reset asynchrone et d'un preset synchrone commandés par un produit global.

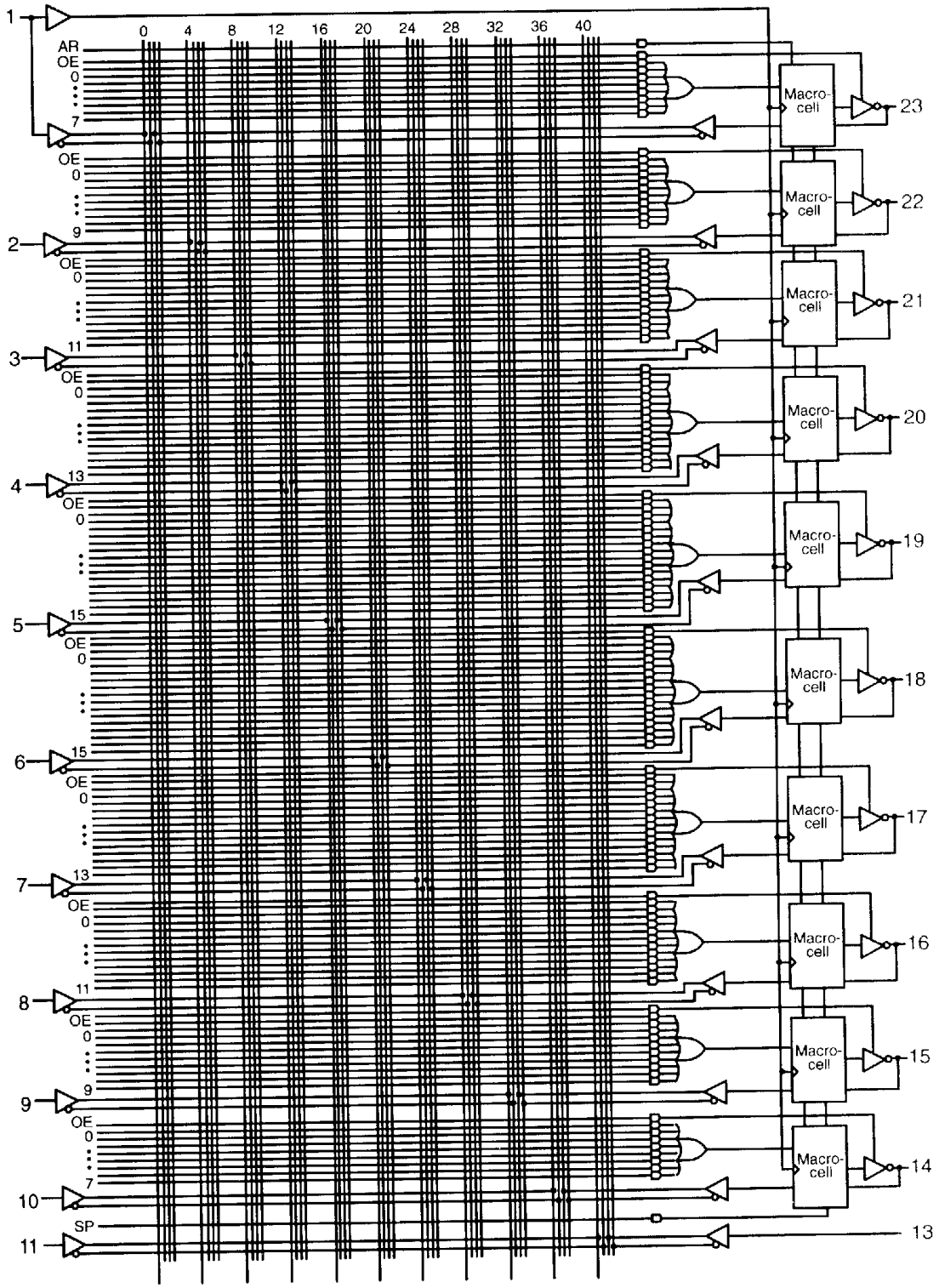


Figure 12 : Structure interne de la GAL22V10

Les data sheets complets relatifs à la GAL22V10 sont donnés en annexe 4.

Manipulations du laboratoire

1 Matériel

Les manipulations de ce laboratoire sont réalisées sur une carte électronique comprenant

- un compteur 4-bit asynchrone (74LS93) avec possibilité de déclenchement sur un appui bouton ;
- 6 inverseurs connectés en cascade (74HC04) ;
- un *timer* (NE555) en montage astable ou monostable ;
- un composant programmable en VHDL (ispGAL22V10), programmée via le connecteur 8 broches.

Ces différents composants possèdent diverses entrées (boutons ou générateur d'ondes) et ont leurs sorties connectées soit à des LEDs soit à un connecteur branché à un oscilloscope via un fil monobrin.

La figure 1 ci-dessous montre le typon, ou schéma PCB, de la carte électronique. La figure 2, quant à elle, est le dessin schématique de la carte. **Prenez un peu de temps pour regarder les connections effectuées et pour en déduire les rôles de chaque composant (circuits intégrés, boutons, LEDs,…) de la carte afin de vous familiariser avec celle-ci.**

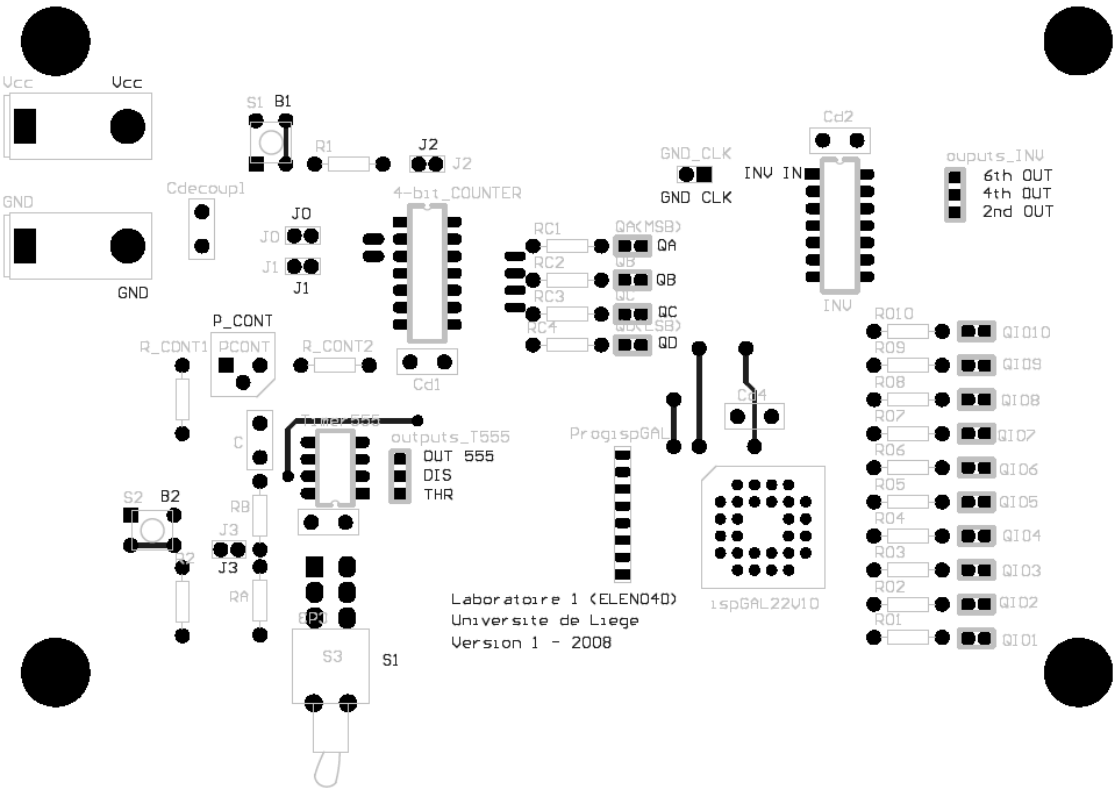


Figure 1 (a) – Typon (schema PCB) de la carte de laboratoire : Couche « top layer » et « top overlay », …

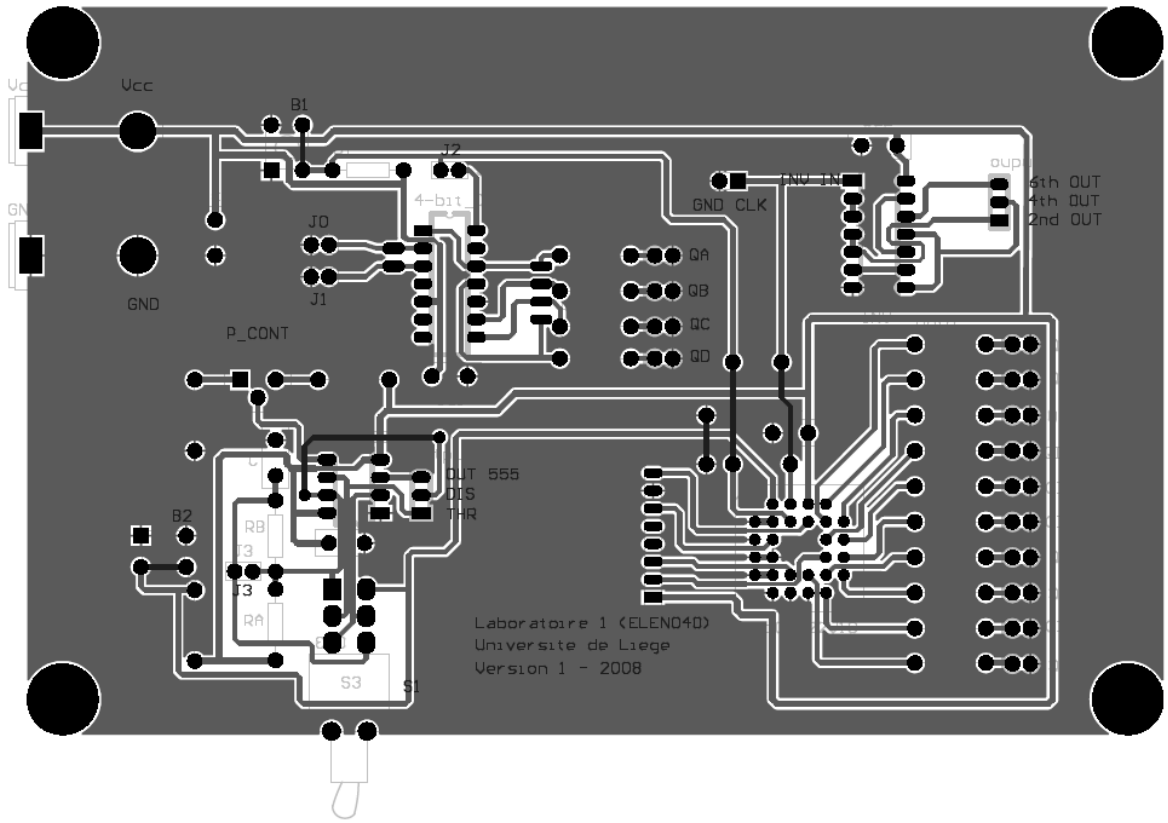


Figure 1 (b) – … ajout de la couche « bottom layer ».

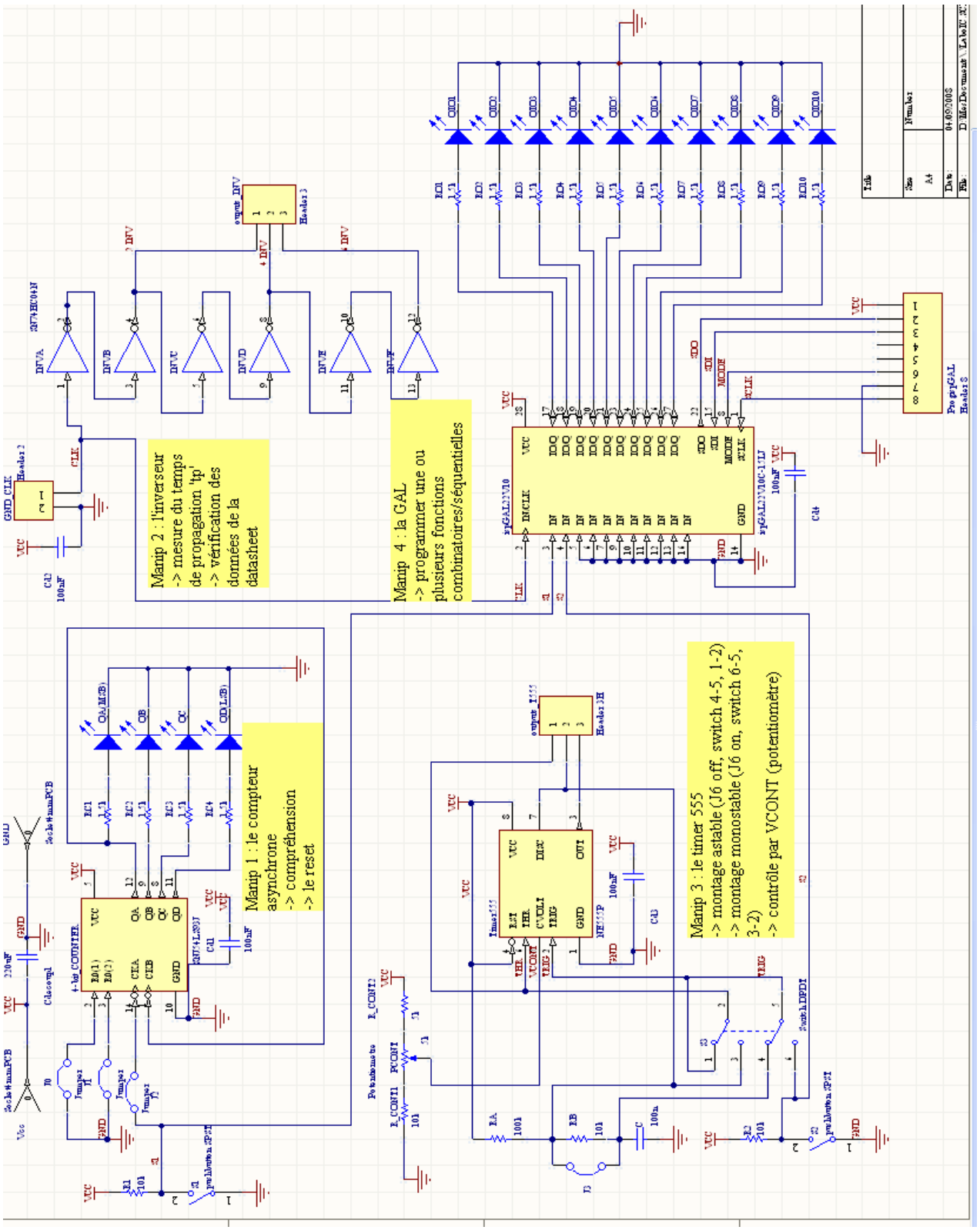


Figure 2 – Schéma de la carte de laboratoire.

2 Le compteur 4-bit asynchrone

Ce chip contient un *Ripple Counter* de 4 bits avec deux broches de *reset*. Le schéma suivant vous montre sa structure interne, externe, sa représentation la plus classique et l'assignation des broches.

Chaque entrée ou sortie reçoit un nom symbolique et un numéro de patte. Plusieurs pattes ne sont pas utilisées dans ce circuit et sont nommées **NC** (no connection). Puisqu'elles sont totalement isolées du reste du circuit, ces entrées peuvent rester flottantes. Sur le schéma-bloc (figure 3), les entrées sont à gauche, les sorties à droite et les alimentations sur les deux côtés restants. Le numéro d'ordre du chip, les noms des pattes et leurs numéros sont reproduits. Le fait de placer les entrées à gauche et les sorties à droite est une convention souvent utilisée pour clarifier les diagrammes logiques : les données se déplacent de gauche à droite, la séquence des événements est donc plus facilement décelable. Les datasheets complets sont fournis en annexe.

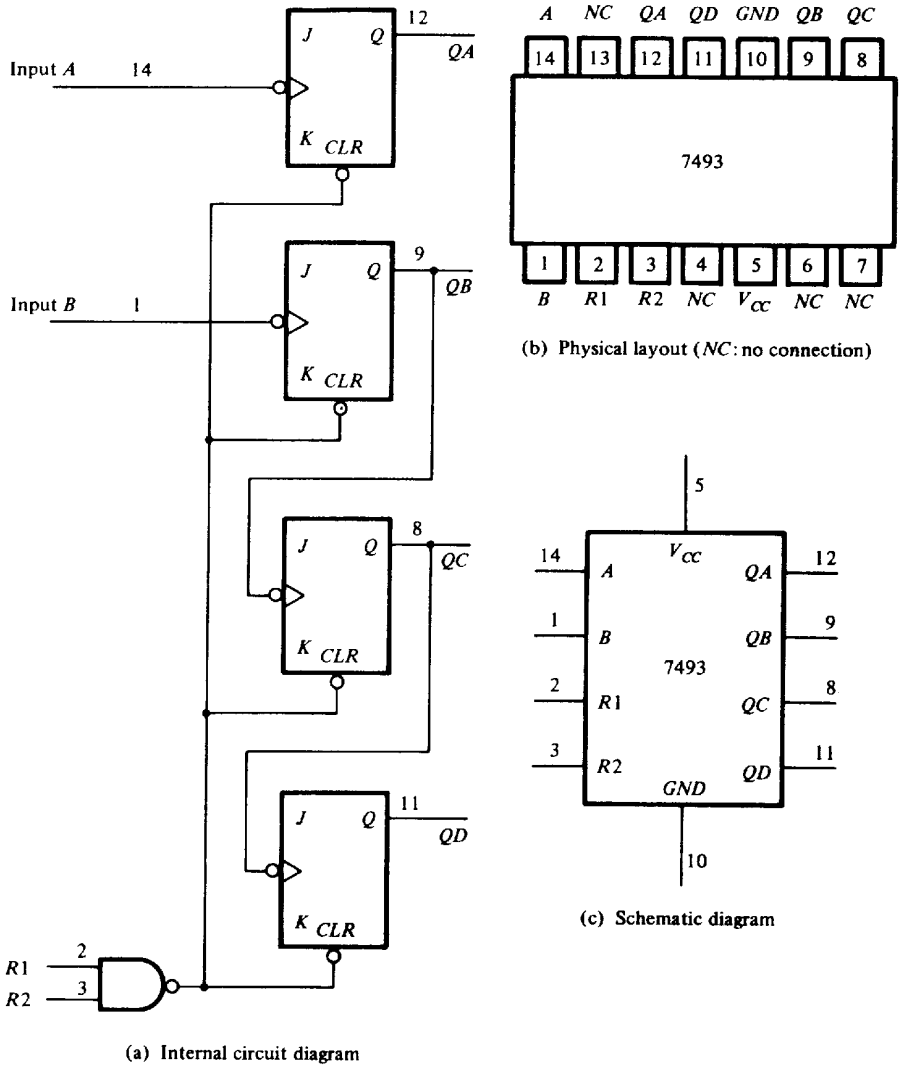


Figure 3 – schéma interne et externe du *4-bit Ripple counter*.

Cette première expérience servira de première approche aux méthodes d'introduction et de lecture des données et aux divers instruments du laboratoire. Nous utilisons le chip décrit ci-dessus, à savoir, le **74HC93** (voir caractéristiques complètes dans les datasheets). Le circuit se compose de quatre flip-flops dont trois sont reliés en chaîne. Pour former un compteur de 4 bits. Les trois premiers points vous aideront à comprendre les connections entre les différentes broches du compteur. Lisez-les attentivement avant de commencer !

a. Vérifiez sur la carte, ainsi que sur le schéma page 2, la connexion entre le 4ème flip-flop et les trois autres, c-à-d entre Q_A et CP1 (=B).

Q_A, Q_B, Q_C et Q_D sont alors les sorties binaires du compteur, avec Q_A comme le bit le moins significatif (**L**east **S**ignificant **B**it) et Q_D, le MSB (**M**ost **S**ignificant **B**it). Chaque flip-flop dispose d'un reset asynchrone commun commandé par les deux entrées MR1et MR2 à travers une porte NAND. D'après les datasheets, on remarque que des '1' sur ces entrées placent le compteur à '0'.

b. N'oubliez donc pas de raccorder MR1 (=R1) et MR2 (=R2) à la masse si vous voulez libérer le compteur via les Jumper J0 et J1.

Deux pattes de remise à zéro sont utiles si vous voulez interrompre le compteur avant la fin naturelle de la séquence (0000, 0001,... , 1110, 1111, 0000, ...).

c. Réalisez toujours vos connexions avec l'alimentation et les entrées coupées. Allumez ensuite l'alimentation et puis les entrées.

Pour éteindre ou modifier le câblage, coupez les entrées, ensuite l'alimentation. Remarque : les broches d'alimentation ne sont pas aux endroits habituels sur ce circuit intégré. Comme première partie de la manipulation, testez le compteur 4 bits. Pour cela, utilisez un générateur d'impulsions branché à l'entrée d'horloge CP0 (=A). Déconnectez le jumper J2 de la carte !

d. Visualisez les sorties du compteur à l'aide de LEDs

e. Utiliser ensuite un interrupteur (bouton poussoir et une résistance tire haut, placez le jumper J2) comme entrée logique. Visualisez l'effet des rebonds sur les LEDs.

Un oscilloscope numérique permettrait de voir le rebond. Sur base de vos observations, esquissez son allure :

f. Dans le cas de l'oscilloscope double trace, comparez les traces de l'entrée d'horloge avec les différentes sorties alternativement. La fréquence des impulsions sera de 10KHz ou plus.

CLK

Q_A

Q_B

Q_C

Q_D

Quel(s) lien(s) existe(nt) entre ces différentes traces ?

Les nombres décimaux codés binaires (BCD) utilisent les nombres binaires de 0000 à 1001 pour représenter les chiffres de 0 à 9. Le 74HC93 peut être utilisé en tant que compteur BCD.

g. <u>Débranchez les jumper J0 et J1</u>. Connectez les entrées MR0 et MR1 à 2 des sorties (QA à QD) à l'aide de fils monobrin de manière à obtenir un compteur de ce type. Auriez-vous procédé de la même manière si vous aviez eu un compteur synchrone. Pourquoi?

	MR0	MR1	Pourquoi ?
Compteur Asynchrone			
Compteur Synchrone			

Le 74HC93 peut également être connecté pour compter de 0 à divers nombres. A quoi faut-il connecter MR0 et MR1 pour obtenir un compteur de 0 à 5?

h. Vérifiez votre compteur BCD en l'implémentant et en le testant.

Un compteur 3 bits est très utile pour générer les entrées des autres schémas logiques. C'est en effet une méthode facile pour vérifier les tables de vérité d'une conception combinatoire puisque le compteur passe par toutes les combinaisons binaires.

3 Délai d'inverseurs en cascade

a. En utilisant l'oscilloscope, on vous demande de déterminer le délai dans un inverseur (utiliser une fréquence d'horloge voisine de 1MHz).

De manière à visualiser un délai plus important, il est conseillé de mesurer le délai de 2, 4 ou 6 inverseurs successifs. Comparer vos résultats avec les valeurs données dans les datasheets. Remarque : Le signal est plus facile à mesurer si vous comparez deux sorties d'inverseurs, par exemple, la sortie du premier et du cinquième inverseur.

Délai de	2 inverseurs	4 inverseurs	6 inverseurs
Datasheet			
Mesuré			

Conclusion :

4 Le timer 555

Le timer 555 est un des IC's les plus utilisés. Il peut notamment servir à générer un pulse de longueur définie suite à une impulsion de quelque longueur que ce soit. C'est ce que l'on appelle le mode "monostable". Il peut aussi être connecté pour produire une série continue de pulses à une fréquence contrôlée. C'est le mode "astable". Les périodes en mode astable sont contrôlées par les valeurs de Ra, Rb et C. La sortie sera haute durant un temps Th donné par :

$$Th = 0.695 \times (Ra + Rb) \times C \ .$$

La sortie sera basse durant un temps Tl donné par

$$Tl = 0.695 \times Rb \times C \ .$$

Avec un peu d'algèbre, il peut être montré que la période totale T et la fréquence F sont données par :

$$T = Th + Tl = 0.695 \times (Ra + 2Rb) \times C \ \text{ et } \ F = \frac{1}{T} = \frac{1.44}{(Ra + 2Rb) \times C}$$

Le 555 possède deux entrées de contrôle additionnelles. Un "reset" arrêtera immédiatement

toute sortie et imposera un niveau bas à la sortie. Le "contrôle voltage" (Cv) peut être utilisé pour faire varier la largeur du pulse de sortie en changeant la composante DC lui étant appliquée, ce qui peut aussi changer la fréquence astable d'oscillation.

a. Placer le timer555 en mode astable (*J3 OFF, switch haut*), Ra = 100k, Rb = 10k et C = 100nF. Connecter le channel 1 de l'oscilloscope et le régler afin de pouvoir observer un ou deux cycles du signal de sortie du 555.

- Connecter le channel 1 de l'oscilloscope et le régler afin de pouvoir observer un ou deux cycles du signal de sortie du 555.
- Connecter le channel 2 de l'oscilloscope tout d'abord sur l'entrée "discharge" et ensuite sur l'entrée "threshold" ou "trigger" qui sont connectées ensemble.
- Dessiner à l'échelle les trois signaux (figue 4). Soyez sûr de bien représenter leurs relations au temps. Insérer les valeurs observées en temps et en amplitude sur le graphe. Expliquez le comportement des courbes (le schéma interne de ce composant est donné à la figure 4 et vous permettra de mieux comprendre son fonctionnement).

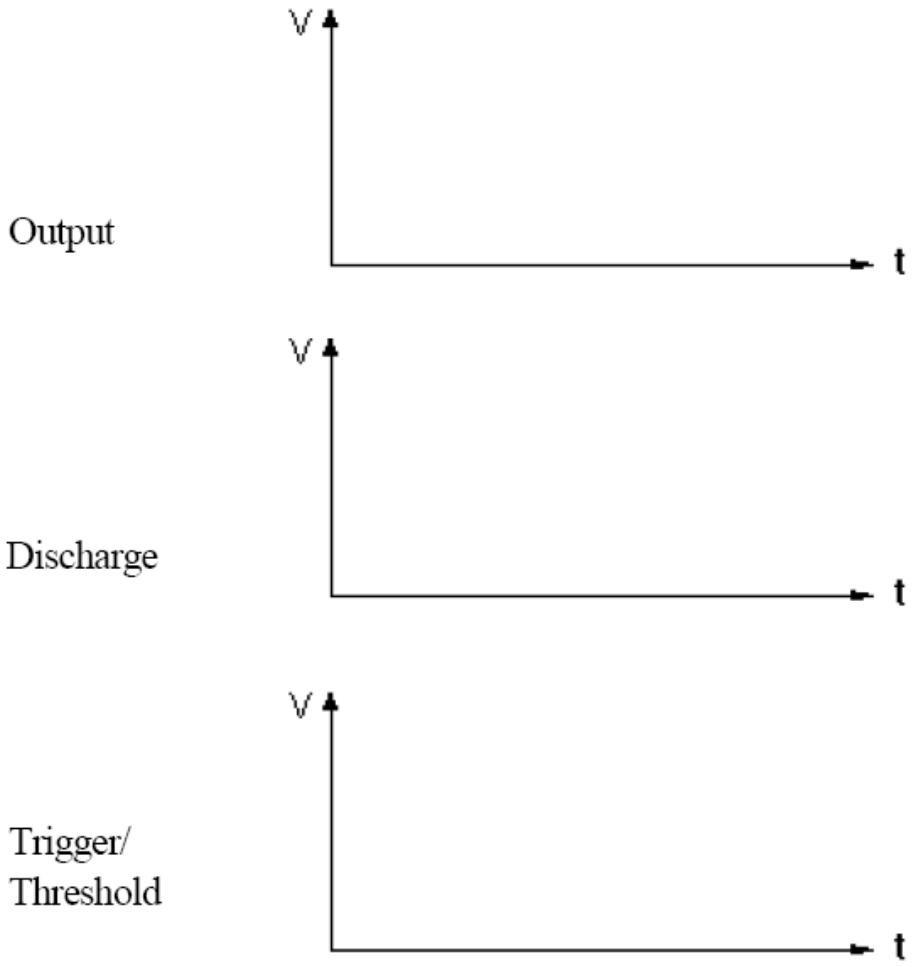


Figure 4 – Complétez ces diagrammes termporels.

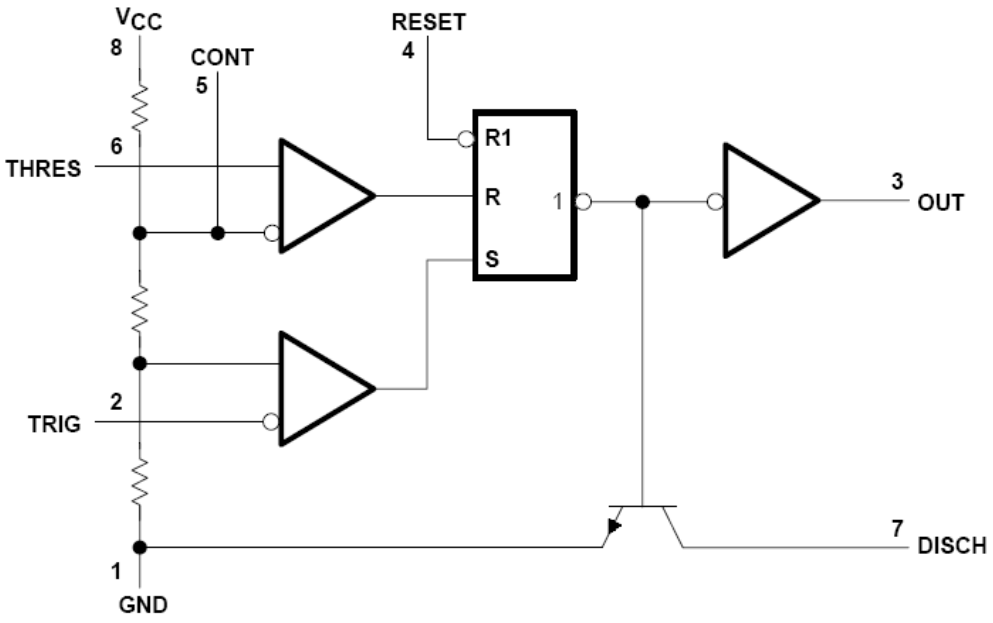


Figure 5 - Schéma interne du 555.

- Tournez le potentiomètre. Expliquez.

b. Placer le timer555 en mode monostable (*J3 ON, switch bas*), Ra = 100k et C = 100nF. Connecter le channel 1 de l'oscilloscope et le régler afin de pouvoir observer un ou deux cycles du signal de sortie du 555.

Le bouton est le signal de commande des pulses de sortie. La largeur du pulse vaut

$$T_w = 1.1\ R_A\ C$$

Qu'observez-vous en appuyant sur le bouton (appui court et long) ?

Le montage fonctionne-t-il comme attendu ?

Que feriez-vous pour vous s'assurer que le montage est bien un montage monostable ?

A quoi sert un tel montage ?

5 Un composant programmable : la GAL

a. Programmer un compteur 3-bit synchrone en VHDL. Utilisez la suite de logiciels *ispLEVER CLassic* de *Lattice*, ainsi que le module USB de programmation HW-USBN-2A. Attention à l'attribution des pins !!! Elle est fixée par la carte !!!

Voici un code VHDL d'un tel compteur, vous le trouverez dans le fichier **D:\Documents\ELEN040\Labo\VHDL\LABOEN.vhd ou C:\ELEN040\VHDL\LABOEN.vhd**

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity LABOEN is port(
      CLK, B1: in std_logic ;
      O: out std_logic_vector ( 2 downto 0 ));

-- attributions des pins selon SYNPLIFY sous ispLEVER
attribute loc: string;
attribute loc of B1: signal is "p3";
attribute loc of O: signal is "p17,p18,p19";

end LABOEN;

architecture ARCH_LABOEN of LABOEN is

signal Q : std_logic_vector(2 downto 0); -- signal interne, FF

begin

  process(CLK)                    -- process compter
  begin
    if CLK'event and CLK='1' then  -- flanc montant de CLK
      if B1='0' then               -- reset synchrone
        Q <= "000";
      else
        Q <= Q+1;                -- incrément de Q
      end if;
     end if;
  end process;

  process(Q)                      -- process : sortie = valeur de Q
  begin
    O <= Q;
  end process;

end ARCH_LABOEN;
```

b. Programmer 2 fonctions F_1 et F_2 en VHDL

Prenez le compteur Q comme entrées A, B, C des fonctions F1 et F2 et ajoutez un process qui définit les sorties F1 et F2 tel que

- F1 (A, B, C) = Σm (3, 6, 7)
- F2 (A, B, C) = Σm (0, 1, 3, 6, 7)

c. Sachant que vous ne possédez que les composants de la page 4, donnez 3 ***autres implémentations*** **de ces 2 fonctions**

-
-
-

Quels sont les **avantages** et les **inconvénients** de chacune d'elle (taille, simplicité,…) ?

d. Programmer une animation lumineuse

Cette dernière manipulation est un problème de synthèse de circuit.

On vous demande de concevoir et de réaliser une animation lumineuse sur 5 ou 6 LEDs. Le *choix de la séquence et le nombre de séquences est laissé libre*. La seule restriction est d'utiliser *uniquement la GAL et de la programmer en VHDL*.

Essayer de faire preuve d'ingéniosité pour simplifier au maximum votre circuit (pensez éventuellement aux don't care si votre circuit en comporte...). De manière à tester votre implémentation, alimentez votre circuit au moyen du compteur réalisé lors de la première manipulation. **Donnez le code VHDL et éventuellement la table de vérité de votre circuit.**

Voici un exemple :

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity animlum is port(
     CLK: in std_logic ;
     B1: in std_logic;
     B2: in std_logic;
     O_CNT: out std_logic_vector(2 downto 0);
     O_ANIM: out std_logic_vector(4 downto 0) );

-- attributions des pins selon SYNPLIFY sous ispLEVER
attribute loc: string;
attribute loc of B1: signal is "p3"; -- voir schema-carte Fig.2
attribute loc of B2: signal is "p4"; -- voir schema-carte Fig.2
attribute loc of O_CNT: signal is "p17,p18,p19";
attribute loc of O_ANIM: signal is "p20,p21,p23,p24,p25";

end animlum;

architecture arch_animlum of animlum is

-- definir ETAT = machine d'états de 2 bits, donc 4 etats
-- 00 : mode stop
-- 01 : mode défilement "gauche"
-- 10 : mode défilement "droite"
-- 11 : non utilisé

-- Déclaration des signaux internes
signal ETAT: std_logic_vector(1 downto 0);
signal S_CNT : std_logic_vector(2 downto 0);
begin

-- process CNT
process(CLK)
begin
  if CLK'event and CLK='1' then
    if S_CNT > "011" then
           S_CNT <= "000"; -- compte de 0 à 4 seulement
    else   S_CNT <= S_CNT +1;
    end if;
  end if;
end process;

-- process ETAT
process(CLK,B1,B2)
begin
  if CLK'event and CLK='1' then
    if B1='0' and B2='0' then ETAT <= "00";
    elsif B1='0' then ETAT <= "01";
    elsif B2='0' then ETAT <= "10";
    else NULL;
    end if;
```

```
  end if;
end process;


-- process ANIM
process(S_CNT,ETAT)
begin
  case ETAT is
    when "00" => NULL;
    when "01" => -- défilement gauche
      case S_CNT is
        when "000" => O_ANIM <= "10001";
      when "001" => O_ANIM <= "00011";
      when "010" => O_ANIM <= "00110";
      when "011" => O_ANIM <= "01100";
      when "100" => O_ANIM <= "11000";
      when others => O_ANIM <="-----"; -- don't care
       end case;
    when "10" => --défilement droite
      case S_CNT is
        when "000" => O_ANIM <= "10001";
      when "001" => O_ANIM <= "11000";
      when "010" => O_ANIM <= "01100";
      when "011" => O_ANIM <= "00110";
      when "100" => O_ANIM <= "00011";
      when others => O_ANIM <="-----"; -- don't care
       end case;
    when others => O_ANIM <="-----";
  end case;
  -- afficher le compteur
  O_CNT <= S_CNT;
end process;


end arch_animlum;
```

Programmation d'un PLD

1 Introduction

Ce chapitre vous indique la marche à suivre pour réaliser efficacement votre PLD à partir d'une description des spécifications du circuit en langage VHDL.

Avant de commencer la description et la réalisation du design, il est vivement conseillé de prendre connaissance des caractéristiques de la GAL22V10 (à savoir, l'architecture de la GAL, le nombre d'entrées, de sorties de registres, …) puisque c'est cette GAL que vous allez utiliser. Ces caractéristiques sont décrites dans les datasheets en annexe 4.

Les notes qui suivent sur le logiciel WARP2 (version 1.5) ne se veulent pas exhaustives, elles ne constituent qu'une aide primaire lors de la phase de familiarisation du logiciel. Si vous désirez plus d'informations concernant les possibilités qu'offre le logiciel, référez-vous au manuels d'utilisation ou de référence fournis avec WARP (deux fichiers Acrobat (*.pdf)).

La suite ispLEVER Classic ne diffère pas de WARP, en ce sens, qu'ispLEVER comprend l'éditeur et le compilateur VHDL (qui génère le fichier JEDEC) auquel il faut joindre HDL-Sim pour la simulation.

2 Le logiciel WARP2 (release 4.3)

2.1 Présentation du logiciel

Le logiciel WARP2, édité par la société CYPRESS, permet, à partir d'une description de votre PLD (ou CPLD) en VHDL (fichier.vhd), de générer le fichier JEDEC qui sert à programmer physiquement le composant sélectionné.

Pour ce faire, Warp2 est constitué de 2 outils :

- ***Galaxy*****,** le programme principal, qui contient :
  - un éditeur VHDL,
  - un compilateur VHDL qui génère les équations de fonctionnement et les optimise,
  - un "fitter" qui tente de placer ces équations dans le circuit sélectionné et génère le fichier JEDEC nécessaire au programmateur et au simulateur,
- ***Nova*****,** qui permet la simulation du code VHDL.

Remarque : Ce document ne donne pas d'explication quant à l'utilisation de ce simulateur. Il faut néanmoins savoir que celui-ci est totalement compatible avec les méthodes vues au cours mais que la simulation est très délicate à réaliser pour des circuits PLD complexes. En effet, les contraintes imposées par l'architecture de ces circuits peuvent amener Warp à "comprendre" le code VHDL de façon différente pour l'intégration dans le circuit et pour la simulation. Il est conseillé de se référer au manuel de référence de Warp pour éviter les pièges de la simulation. L'utilisation de Nova est simple et ne devrait pas vous poser de problèmes.

2.2 Utilisation de Galaxy

Lors du lancement de Galaxy, le programme vous demande de spécifier le nom du projet en cours (fichier *.wrp). Apparaît alors la fenêtre de projet. Vous devez commencer par spécifier les fichiers VHDL (*.vhd) qui entreront dans la compilation, par la commande *File - Add*. Une fois les fichiers définis, vous pouvez les éditer en les sélectionnant et en utilisant le bouton *Edit*.

La compilation, suivie de la phase de "fitting", est invoquée par le bouton *Smart*. Il est évidemment indispensable d'ajuster les paramètres au préalable (voir ci-dessous). Etant donné que l'on peut utiliser plusieurs fichiers VHDL, il faut avant de lancer la compilation, spécifier le fichier "top", contenant l'entité principale, par le bouton *Set Top*.

Après une compilation, réussie ou non, il est souvent bon de visualiser le fichier "report" associé, qui indique les différentes phases ainsi que les succès et échecs éventuels. Pour ce faire, sélectionner le fichier (souvent le fichier "Top") et sélectionner la commande *Info - Report File*.

Une fois la compilation terminée avec succès, Warp génère le fichier JEDEC (*.jed) qui servira au programmateur et au simulateur Nova.

2.3 Ajustage des paramètres de synthèse

Avant de lancer le compilateur, certains paramètres doivent être ajustés. Pour les applications que vous réaliserez, régler :

- Dans : *"Generic ..."* :
    - *Optimization* à Exhaustive
    - *Goal* à Area
    - Cochez la case Detailed Report

  Laissez les autres options inchangées (Voir l'aide en ligne pour plus de détails).
- Dans : . "Device ..." :
    - *Device* : C22V10
    - *Package* : PALCE22V10-7PC
    - *Unused outputs* **: état de sortie des broches I/O non-utilisées (Z est un bon choix)**
    - *Post-JEDEC Sim* : **si vous faites de la simulation, vous utiliserez certainement "1164/VHDL", sinon laissez à "none".**
    - *Tech Mapping* : **laissez ces paramètres inchangés car ils n'interviennent que dans des cas très spécifiques d'optimisation difficile.**

2.4 Implémentation de machines d'état en VHDL

Vous trouverez ci-dessous un exemple typique de code pour une machine d'état de Mealy. La machine de Moore n'en est qu'un cas particulier.

Il peut paraître surprenant de définir les sorties combinatoires du système séparément des équations de transition entre états. En fait, le compilateur a très vite tendance à rendre séquentiels des signaux combinatoires si on essaie de les assigner directement dans le process de transition.

La structure du process de transitions ne doit pas être changée pour garder le caractère **synchrone** des changements d'état et maintenir le reset **asynchrone**, car il n'est pas possible de faire autrement avec les circuits proposés.

Exemple :

```
library ieee;
use ieee.std_logic_1164.all;


entity mealy_machine is port( clk, rst:  in std_logic;
                              id:        in std_logic_vector(3 downto 0);
                              w:         out std_logic;
                              y:         out std_logic_vector(1 downto 0));
end mealy_machine;



architecture arch_mealy_machine of mealy_machine is
      type states is (state0, state1, state2, state3, state4);
       signal state: states;
begin
transitions: process (clk, rst)
            begin
             if rst='1' then
                  state <= state0;
             elsif (clk'event and clk='1') then
                  case state is
                        when state0 =>
                              if id = x"3" then
                                    state <= state1;
                              else
                                    state <= state0;
                              end if;
                        when state1 =>
                              state <= state2;
                        when state2 =>
                              if id = x"7" then
                                    state <= state3;
                              else
                                    state <= state2;
                              end if;
                        when state3 =>
                              if id < x"7" then
                                    state <= state0;
```

```
                              elsif id = x"9" then
                                    state <= state4;
                              else
                                    state <= state3;
                              end if;
                        when state4 =>
                              if id = x"b" then
                                    state <= state0;
                              else
                                    state <= state4;
                              end if;
                  end case;
            end if;
end process;

--assign moore state outputs;
y <= "00" when (state=state0) else
  "10" when (state=state1 or state=state3) else
  "11";

--assign mealy output;
w <= '0' when (state=state3 and id < x"7") else
  '1';
end arch_mealy_machine;
```

Si on le désire, il est possible dans le code VHDL de spécifier **l'assignation des broches**. Celle-ci se fait via une directive placée dans la déclaration d'entité, juste après la définition des entrées et sorties.

Exemple :

```
entity mealy_machine is port(
             clk, rst:  in std_logic;
             id:        in std_logic_vector(3 downto 0);
             w:         out std_logic;
             y:         out std_logic_vector(1 downto 0));

attribute pin_numbers of mealy_machine:entity is
             "clk:2" &
             "rst:3" &
             "id(0):8" &
             "id(1):9" &
                 ...

             "y:5";
end mealy_machine;
```

Si certaines ou toutes les broches n'ont pas de numéro assigné, le "fitter" choisira une solution (pas toujours optimale). Il faut en outre savoir que l'assignation forcée par un attribut pin_numbers a certains inconvénients au niveau de l'optimisation et peut conduire à une impossibilité de fitting.

Dans le cas de la 22V10, le nombre de termes dans les sommes logiques à l'entrée des cellules n'est pas le même pour toutes les cellules et varie de 8 à 16. Si une erreur intervient au fitting, il peut alors être nécessaire de modifier les assignations.

2.5 Quelques remarques

Il est parfois utile, pour simplifier la structure du projet, de décomposer l'architecture de l'entité principale en plusieurs sous-entités représentées en VHDL par des "components". Cette technique présente toutefois l'inconvénient suivant.

Pour pouvoir interconnecter ces composants à l'intérieur de l'entité, il faut créer des signaux auxquels on fera référence dans l'instanciation des composants. Hélas, le compilateur impose de sacrifier une cellule pour générer l'équation logique d'un signal. Si celui-ci constitue la sortie séquentielle d'un composant, c'est acceptable. Mais si le signal est connecté à une sortie purement combinatoire, il y aura gaspillage d'une cellule.

En résumé, la subdivision à outrance du design est à éviter, sous peine d'être rejetée par le "fitter".

Rappelons que sur la GAL22V10 :

- une seule broche peut être utilisée comme horloge,
- l'utilisation d'un flip-flop condamne la broche associée, même si vous n'avez pas de nécessité à voir apparaître la sortie du flip-flop à l'extérieur du circuit intégré,
- les flip-flops sont obligatoirement de type D,
- les signaux SET et RESET sont communs à tous les flip-flops.

3 Travail à fournir

Plusieurs fichiers d'exemples sont fournis avec le logiciel. Vous pouvez vous en inspirer pour comprendre le style de conception et le genre de problèmes généralement résolus par cette méthodologie.

Avant de vous lancer dans la résolution de votre problème, nous vous demandons de venir nous proposer votre idée de manière à s'assurer de la faisabilité du travail. Si d'un autre côté, vous n'avez pas d'idée originale à proposer, nous pouvons toujours vous en fournir une.

Le problème étant choisi, votre travail consistera à trouver une solution efficace et à l'implémenter sur une GAL22V10 (ou plusieurs si votre solution nécessite un nombre important d'entrées/sorties et de termes de produits). Ce composant étant néanmoins très complet, la gamme des problèmes qui peuvent être résolus sur une seule PAL est très vaste. Nous vous demanderons donc d'utiliser au mieux les fusibles. Rien ne sert d'utiliser un composant programmable onéreux pour implémenter une équation logique simpliste. Plus votre chip est "rempli", plus on a intérêt à l'utiliser en remplacement de nombreux composants plus simples.

Une fois votre conception simulée et testée par WARP, vous pouvez passer à la programmation réelle de la GAL. Dans ce but, nous vous conseillons d'acheter une GAL (GAL22V10 ou PALCE22V10) plutôt qu'une PAL car cette première est d'une part, moins onéreuse et d'autre part, elle est effaçable et reprogrammable. Le programmateur de PLD se trouve au R100. Lorsque la programmation est réussie, il vous reste à tester votre chip électriquement. Pour cela, plusieurs possibilités sont envisageables :

- la plus rapide est d'utiliser un breadboard sur lequel vous assemblez vos différents composants. C'est une méthode rapide mais attention aux faux contacts !
- pour les amateurs de fer à souder, vous pouvez monter votre circuit sur une carte prétrouée (disponible dans tous les magasins d'électronique). Cette méthode, si les soudures sont parfaites, est souvent plus fiable que le breadboard.

Remarque : un conseil utile pour cette solution : ne soudez pas directement vos composants sur la plaque mais placer tous vos CI sur des supports

- Pour les plus courageux, un circuit imprimé peut être réalisé. Si vous êtes intéressés, contacter un assistant.

Quelque soit la solution retenue, tous les essais peuvent s'effectuer au R100. Lorsque tous les travaux seront présentables, une journée de démonstration sera organisée.

L'appréciation du travail dépendra principalement du problème choisi (originalité, utilité, complexité ...) et de votre façon de le résoudre (approche du problème, clarté et validité des équations, simulation maximale du circuit...).

Bibliographie

Advanced Micro Devices, Monolithic Memories, *"Pal Device Data Book"*, 1988.

Buchanan J., *"CMOS/TTL Digital Systems Design"*, Mc Graw-Hill Publishing Company, 1990.

Buchla D., *"Digital Experiments Emphasizing Systems and Design"*, Merril Publishing Company, 1986.

Lala P.K., "*PLD : "Digital System Design Using Programmable Logic Devices"*, Prentice/Hall, 1990.

Lancaster D., *"TTL Cookbook"*, Sams Publications, 1977.

Mano M.M., *"Digital Design"*, Prentice/Hall, 1984.

Motorola, *"High-Speed CMOS Data"*, 1993.

Philips S&I Equipment Div., *"PM3215 Operating Manual"*, 1983.

Skahill Kevin, « *VHDL for programmable logic* »,1996

Tocci R., *"Circuits numériques"*, Dunod1992.

Wakerly J.F., *"Logic Design Projects"*, John Willey and sons, 1976.

Annexe 1 : Alimentation stabilisée

1 Structure interne

Les circuits CMOS peuvent être alimentés par une tension comprise entre 2 et 6 volts. Dans les manipulations du laboratoire, nous allons les alimenter par une tension de 5V stable. Pour ce faire, nous allons utiliser l'alimentation stabilisée qui est disponible sur les tables de laboratoire.

Pour le lecteur intéressé, voici quelques explications concernant le structure interne de l'alimentation stabilisée.

Le circuit régulateur le plus courant que l'on trouve dans les alimentations stabilisées est le **7805** qui régule précisément une tension de sortie de 5V et ce, par utilisation d'un seul circuit et pour un coût modeste. La tension à l'entrée doit être supérieure d'au moins 2,5V à celle de sortie; elle peut s'élever jusqu'à 12V. Si le circuit alimenté consomme plus de 100mA, il faudra prévoir un radiateur de refroidissement. Le régulateur limite automatiquement le courant délivré à 750mA et est protégé contre les court-circuits.

Les bornes d'entrée doivent être shuntées par un condensateur de 330nF, celles de sortie nécessitent un condensateur de 100nF. Il doit être le plus près possible des broches de sortie, coupez donc les pattes du condensateur au plus court.

2 Condensateurs de découplage

Pour éviter que des pics de courant ne se propagent vers l'alimentation et les autres circuits, des condensateurs répartis près des chips sont utilisés pour fournir l'énergie demandée pendant les transitions d'état : **despiking capacitors**. Des condensateurs céramiques de 10 à 100 nF avec des broches les plus courtes possible sont recommandés.

Il ne faut pas hésiter à inclure assez de condensateurs : ils corrigent bien souvent des dysfonctionnements incompréhensibles! Il est conseillé d'en placer un par chip ou au minimum un pour 4 chips SSI, ou un pour 2 MSI. De manière générale, pour qu'un condensateur agisse sur un circuit, il doit être à moins de 5 cm de celui-ci.

Un condensateur de 10 µF, 6V (ou plus) électrolytique au tantale stabilise localement l'alimentation de chaque circuit imprimé.

Annexe 2 : Timer 555

Ce circuit analogique est très utilisé car il permet, très simplement et à peu de frais, de réaliser diverses fonctions avec précision : génération d'ondes carrées *(astable multivibrator)*, source de signal, génération de pulses de durée calibrée *(monostable multivibrator),…* Connecté à une alimentation de 5V, ce circuit est compatible TTL. Il peut aussi être utilisé avec des tensions de 4V à 15V et peut fournir *(source)* ou accepter *(sink)* un courant de plusieurs centaines de mA.

1 Diagramme interne

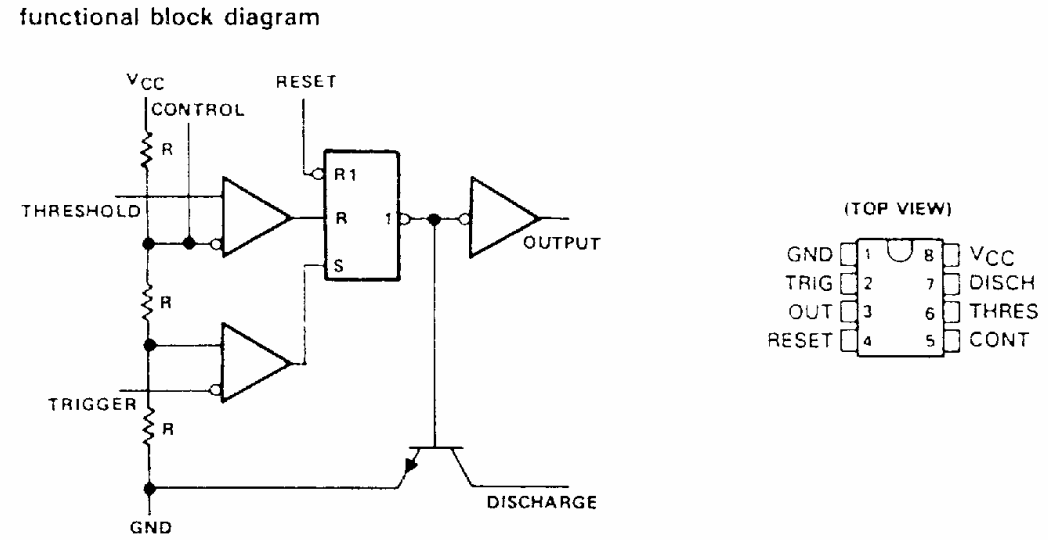


Figure 13 : Fig. tirée de Texas Instruments Catalog, Texas Instruments, 1989

Les deux triangles sont des comparateurs : on peut considérer que si la tension sur la borne négative (avec une bulle) est inférieure à la tension sur la borne positive, la sortie est logiquement vraie.

Deux des entrées des comparateurs sont reliées de façon interne sur un diviseur de tension. La tension VCC-GND est divisée en trois par des résistances. Nous n'utiliserons pas la borne CONTROL qui peut modifier cette division. En TTL, les tensions de déclenchement (TRIGGER et THRESHOLD) sont donc de 1/3*5 = 1,67V et 2/3*5 = 3,33V.

Les sorties des comparateurs activent un flip-flop RS avec RESET. La sortie Q est disponible pour l'utilisateur, c'est elle que nous utiliserons. Q' commande un transistor qui laissera passer le courant entre les broches DISCHARGE et GND si elle est haute (Q'=1 => Q=0).

Pour utiliser ce chip, il faut l'entourer de composants externes (circuit(s) RC).

2 Réalisation d'un générateur d'impulsions

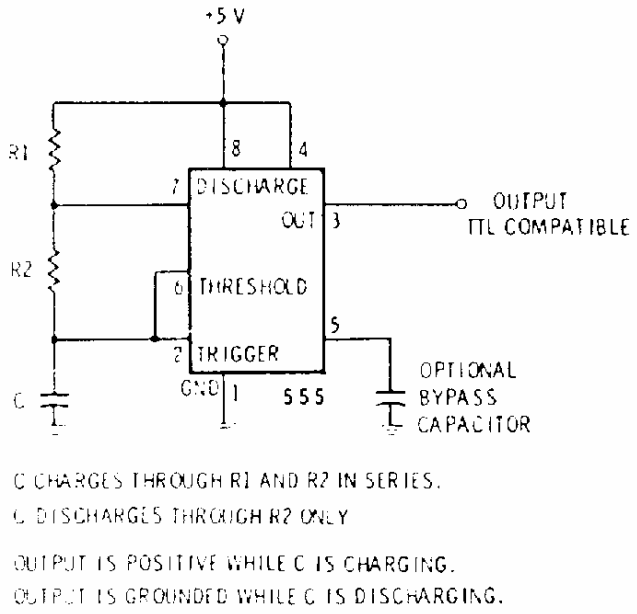


Figure 14 : Fig. tirée de Texas Instruments Catalog, Texas Instruments, 1989

Supposons que la sortie est active (1), le condensateur déchargé, le transistor de décharge est coupé. Le condensateur se charge à travers les deux résistances en série vers +5V. Quand la tension sur C dépasse 2/3 de VCC, le comparateur du haut envoie un 'reset' au flip-flop. La sortie devient basse et le transistor s'ouvre, laissant C se décharger à travers la résistance R_2 (ou R_b). La décharge est effective jusqu'à ce que la tension de C tombe au-dessous de 1/3 VCC. Le deuxième comparateur envoie alors une impulsion 'set' au flip-flop. On est revenu dans le cas de départ.

La tension aux bornes du condensateur oscille donc entre 1/3*VCC et 2/3*VCC, la sortie alterne en onde périodique. Les temps haut et bas sont respectivement :

$$t_H = 0.693.(R_A+R_B).C$$

$$t_L = 0.693.R_B.C$$

Les résistances peuvent varier de 1KW (R_A ou R_B) à 3.3 MW (R_A+R_B). Le condensateur sera supérieur à 1nF. Une bonne temporisation peut être obtenue jusque 330kHz.

On peut jouer sur les résistances pour obtenir un rapport cyclique *(duty cycle = $T_H/(T_H+T_L)$)* variant de 50% à 99.9%. Si $R_B >> R_A$, on obtient une onde quasi-carrée (exemple : 1K et 1M => 0.1% de différence entre les périodes haute et basse). La période haute est toujours plus longue que la période basse.

Remarquons que le schéma utilisé est indépendant de la source de tension sur une longue période grâce à la compensation des effets des variations sur les tensions de référence. La stabilité en température est aussi excellente.

En conclusion, pour une précision inférieure à 1%, ce circuit est idéal si la fréquence n'est pas trop importante. Sinon, il faut utiliser un schéma basé sur un cristal de quartz.

Les datasheets complets des timers 555 sont fournis ci-après :

http://www.p8s.com/circuit/lj/SLFS022.PDF

Annexe 3 : Liste et schémas des ICs

Liste tirée du CMOS, Circuits Intégrés, ECA,96 /97

Schémas tirés de :

- Data book *"high-speed CMOS logic familly",* Philips, 1988
- Data book *"high-speed CMOS data"*, Motorola, 1993
- http://www.semiconductors.philips.com/

Remarque : Si vous devez acheter des ICs pour une application, il est conseillé de ne prendre en compte que les circuits de la famille 74XX En effet, les circuits appartenant aux autres familles, telles les 74XXX et surtout les 74XXXX sont souvent très chers et difficilement disponibles.

Annexe 4 : Liste et schémas des Pals

Liste tirée de PAL Device Handbook, Advanced Micro Devices, 1988

Data sheets tirés de PAL Device Data Book and Design Guide, Advanced Micro Devices, 1995

http://eet.etec.wwu.edu/etec373/AMPAL22V10.pdf

http://www.ece.vt.edu/cel/datasheets/22v10.pdf

http://www.latticesemi.com/products/cpldspld/gal.cfm?source=topnav page des GAL chez Lattice

Annexe 5 : L'oscilloscope

Annexe 6 : Analyseur logique

1 Introduction

Le laboratoire dispose normalement de 3 analyseurs logiques de type Philips PM 3632 muni de sondes 8860.

Ce puissant appareil est précieux pour l'analyse de circuits logiques séquentiels. Son but principal est d'enregistrer une séquence d'états logiques sur plusieurs lignes et de les restituer visuellement sur son écran.

Le nombre maximal de lignes analysées en parallèle est de 32. Si seulement 4 lignes sont nécessaires, la fréquence d'échantillonnage peut s'élever pour atteindre 100 MHz. Le nombre maximal d'échantillons enregistrés dépend du nombre de lignes d'entrées car les données sont mémorisées dans une mémoire de taille fixe. En 32 bits, la 'profondeur' d'enregistrement est de 1000 mots.

2 Présentation de l'analyseur

En gros, un analyseur logique est une sorte d'oscilloscope à mémoire perfectionné, spécialisé dans le traitement des signaux digitaux. Comme tout oscillo à mémoire, il peut déclencher (ou "*trigger*") sur un événement fugitif. Dès que vous appuyer sur **START**, sa mémoire se remplit et se vide perpétuellement à la manière d'une pile puis lorsqu'il rencontre la séquence sur laquelle il doit trigger, sa mémoire accumule les données. Vous pouvez aussi l'obliger à déclencher sur une combinaison de mots (ex : 43H) et stopper l'enregistrement sur une autre (ex : 7FH), à condition toutefois que sa capacité mémoire le permette!

Un conseil : n'hésitez pas à "jouer" avec l'éditeur de l'analyseur, en effet, celui-ci a été conçu d'une manière pédagogique simplifiant assez bien son utilisation.

2.1 Description des touches de l'analyseur

Sur le panneau central de l'analyseur, on trouve les touches suivantes :

- Les touches **START** et **STOP** : pour commencer et arrêter l'acquisition de données.
- Les touches **hexadécimales** : pour modifier les paramètres d'enregistrement et d'affichage.
- Les 4 touches **fléchées** : pour déplacer le curseur sur l'écran.
- Les touches **F1 ->F4** : touches dont la fonction dépend du mode dans lequel on se situe.
- Les 8 touches de programmation (**STATUS**, **CLOCK, TRIG, CONFIG, FORMAT, SEQ, DELAY, SPECIAL FUNCTION**) : pour accéder aux écrans de programmation en vue de spécifier les paramètres destinés à l'enregistrement des données et à la mise en format de l'affichage des données (voir la signification plus loin).
- Les 3 touches d'affichage (**TIMING, STATE, DISA**) : pour définir la structure de l'affichage (voir plus loin).

2.2 Sonde et capteurs

Les signaux provenant du système testé sont toujours transmis à l'analyseur par l'intermédiaire d'une sonde. La configuration de l'analyseur est déterminée automatiquement par la sonde connectée. Il existe plusieurs sortes de sondes; celle que vous utiliserez au labo possèdent 42 fils (16 sur un jeu de capteurs et 26 sur l'autre).

De manière à éviter de mélanger tous les fils, vous pouvez déconnecter le jeu de 26 capteurs car nous n'analyserons jamais plus de 8 signaux simultanément.

Chaque capteur se compose de trois parties :

- le corps
- le capot
- le fil

Le jeu de 16 capteurs comporte :

- **1 capteur d'horloge** (**rouge**) qui doit être connecté à l'horloge du circuit que vous testez.
- **1 capteur de qualification** (**bleu**).
- **6 terres** (**noires**) dont une doit être reliée à la masse du circuit.
- **8 capteurs de signaux**. Ceux-ci ont tous un capot gris et sont identifiés par l'analyseur par un numéro correspondant à leur code couleur (même code que les résistances).

Vous pouvez obtenir le code couleur en sélectionnant la touche **S. FUNCTION** puis 2. Par exemple, le capteur dont le capot est gris, le corps noir (=0) et le fil bleu (=6) correspond à la sonde n° 06. Ces capteurs de signaux doivent être placés à la sortie du circuit intégré (c-à-d avant les résistances) de manière à transmettre une valeur élevée de la tension.

Attention : *Ne jamais brancher, ni débrancher la sonde lorsque l'appareil est sous tension.*

3 Programmation des paramètres

Avant de visualiser vos signaux, plusieurs réglages doivent être réalisés sur l'analyseur. Pour cela, utilisez les touches de programmation : **CLOCK, TRIG, CONFIG, FORMAT, SEQ, DELAY.** Chacune de ces touches permet d'activer un écran offrant des sélections de paramètres et/ou des informations relatives à la programmation. Pour modifier ceux-ci, amenez le curseur dessus et entrez la fonction désirée. La touche **STATUS** sélectionne l'affichage d'état qui résume les données relatives aux paramètres de programmation. La touche **SPECIAL FUNCTION** ne vous servira que pour visualisez le code couleur des résistances **(S FUNCTION 2**)

Les paramètres que vous devez absolument modifier sont les suivants :

3.1 Format d'affichage (touche FORMAT)

La première opération à effectuer est de sélectionner **LOG4** (ou **LOG8**) car d'une part nous utiliserons jamais que 8 sondes au maximum et d'autre part cela remplace tous les autres paramètres utilisés dans les autres écrans de programmation par des valeurs par défaut.

Deux modes de format d'affichage apparaissent, à savoir :

- Affichage temporel (**TIMING**) : les données sont visualisées sous forme d'ondes. Indiquez dans chaque case (12 au max) le numéro du capteur que vous voulez examiner. Des lignes vides ("XX") peuvent être utilisées pour laisser des espaces.
- Affichage d'état (**STATE**) : pour chacune des 8 colonnes, il y a 4 sélections possibles.
    - la base utilisée (bin, hex, dec, ...)
    - la voie la plus haute ('high probe')
    - la voie la plus basse ('low probe')

Remarque : Par défaut, les sondes sont regroupées par groupe de 4.

- la logique (pos. ou neg.)

3.2 Ecran de configuration (CONFIG)

La largeur d'enregistrement est directement mise à jour lorsque vous modifier l'écran format. Comme vous n'utilisez pas de qualification, laissez les autres paramètres sur off.

3.3 Programmation de l'horloge (CLOCK)

Cet écran permet de sélectionner les paramètres relatifs à l'horloge.

- Horloge externe/interne : ici, nous travaillerons toujours en **horloge externe** afin de nous synchroniser sur notre circuit.
- Flanc d'horloge : vous pouvez laisser la valeur par défaut.
- Qualificateur d'horloge : placez le sur always afin qu'il soit toujours validé. En effet, les horloges externes ne sont reconnues que si le qualificateur est validé.

3.4 Programmation du déclenchement (TRIG)

Il est possible de définir 4 mots (étiquetés A, B, C, D) qui serviront de déclenchement . Sur l'écran apparaissent 2 lignes pour chaque mot; les mots sont en effet présentés dans la base sélectionnée et en binaire (en dessous). Si vous laissez les valeurs "XXXX" mises par défaut, le déclenchement aura lieu au moment où vous appuyer sur la touche **START**.

3.5 Programmation de séquence de déclenchement (SEQ)

Permet de définir le moment du déclenchement (c-à-d le moment à partir duquel les résultats seront affichés à l'écran). Si aucune séquence n'a été définie dans **TRIG**, cet écran n'a aucun intérêt puisque le déclenchement se fera dès que la touche **START** aura été sélectionnée. Si, par contre, les mots A, B, C, D ont été définis, **SEQ** vous permet de déterminer la condition de déclenchement (A, A+B, ...).

3.6 Programmation du délai (DELAY)

Permet de définir des délais d'enregistrement avant et après le déclenchement. Pour vos manipulations, vous pouvez laisser les valeurs par défaut.

Une fois tous ces paramètres réglés, il ne vous reste plus qu'à commencer l'acquisition. Appuyez pour cela sur la touche **START**.

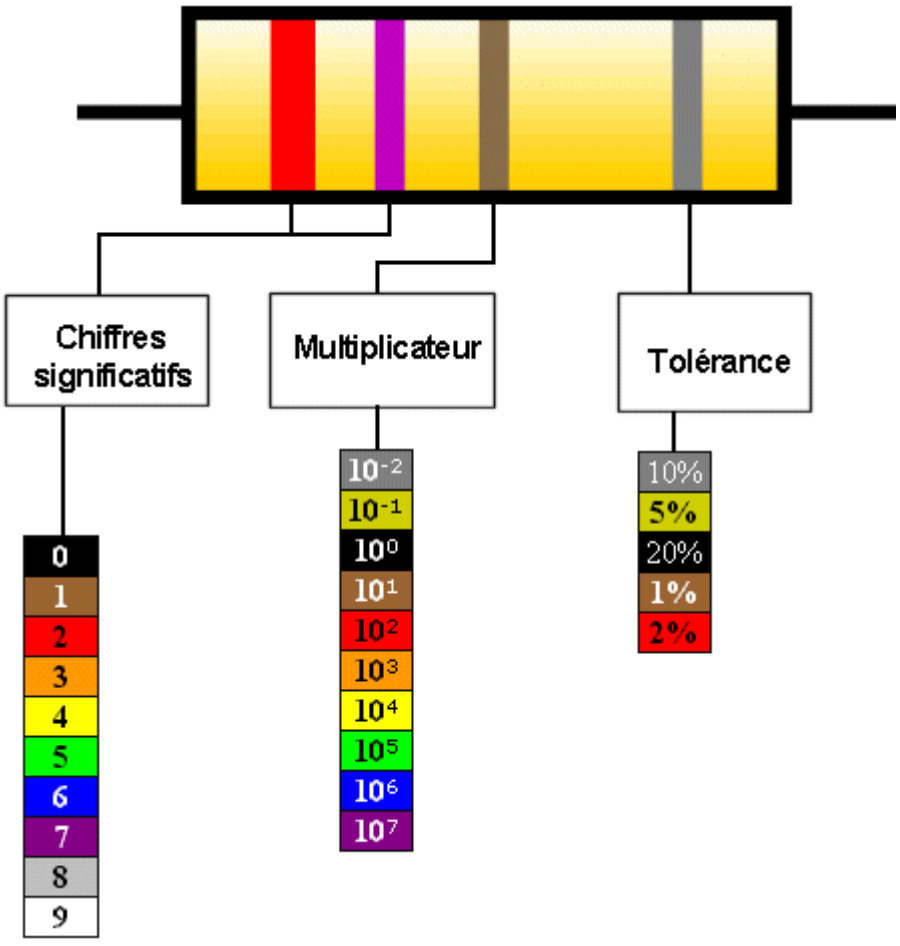
Annexe 7 : Les résistances

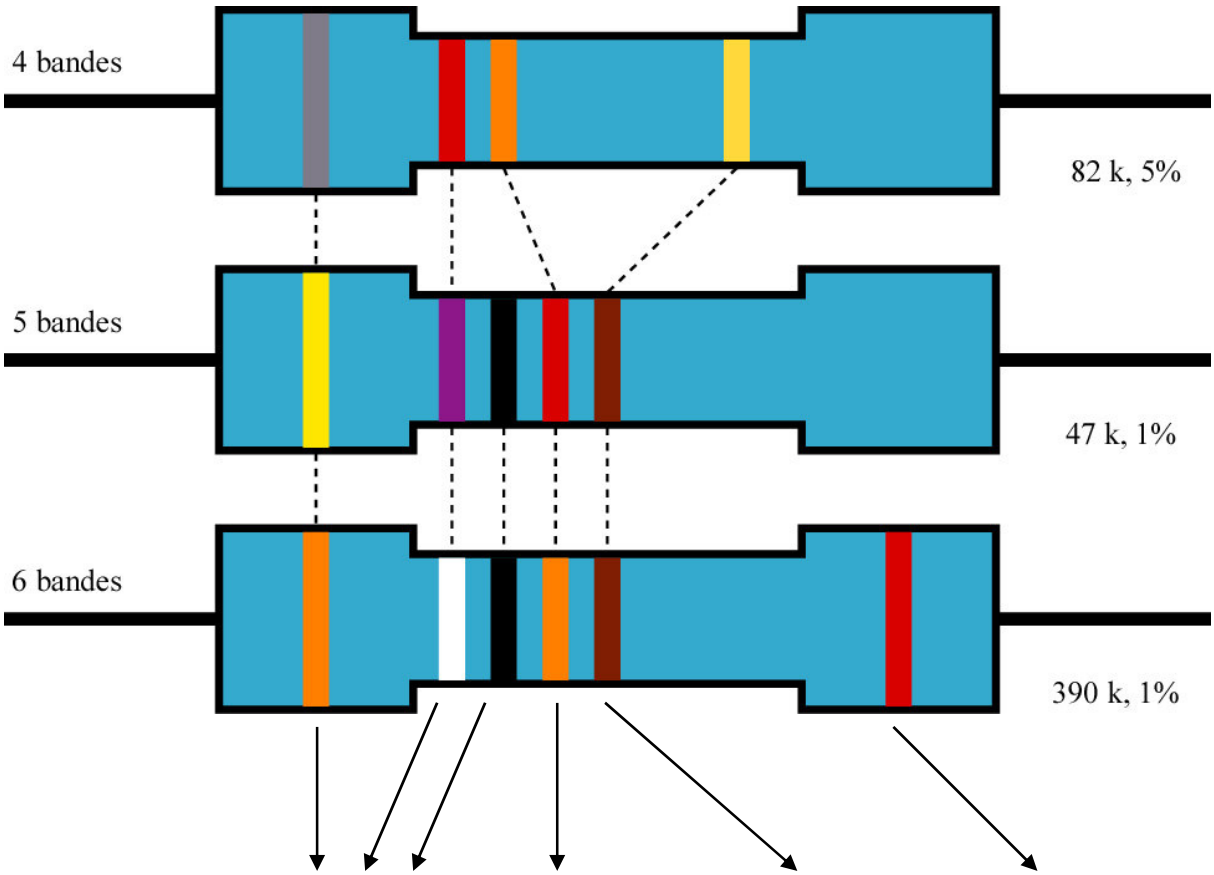
On distingue des résistances avec 4 , 5 ou 6 bandes de couleurs.

On commence par tenir la résistance horizontalement devant soi, et s'il y a un plus grand espace non marqué, on le met à droite ; c'est le cas des résistances à 4 ou 5 bandes.

Pour les résistances à 6 bandes, il est un peu plus difficile de trouver quel est le bon côté, mais on remarquera que l'espace entre les bandes n'est pas identique.

- si la résistance comporte 6 anneaux de couleurs, alors l'anneau le plus à droite indique le coefficient de température,
- l'anneau suivant indique la tolérance,
- l'anneau qui précède indique un multiplicateur (sous forme de 10 exposants quelque chose)
- les 2 ou 3 anneaux à gauche indique la valeur





	valeur	multiplicateur	tolérance	coeff. de t°
(rien)			± 20%	
argent		x 0,01	± 10%	
or		x 0,1	± 5 %	
noir	0	x 1		200 10^{-6}
brun	1	x 110	± 1 %	100 10^{-6}
Rouge	2	x 100	± 1 %	50 10^{-6}
Orange	3	x 1 k		15 10^{-6}
Jaune	4	x 10 k		25 10^{-6}
Vert	5	x 100 k	± 1 %	
Bleu	6	x 1 M	± 1 %	
Violet	7	x 10 M	± 1 %	
Gris	8			
Blanc	9			

Ne	**Mangez**	**Rien**	**Ou**	**Jeûnez**	**Voilà**	**Bien**	**Votre**	**Grande**	**Bêtise**
Noir	**Marron**	**Rouge**	**Orange**	**Jaune**	**Vert**	**Bleu**	**Violet**	**Gris**	**Blanc**
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**